NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

INFRARED FACE RECOGNITION

by

Colin K Lee

June 2004

Thesis Advisor:	Monique P. Fargues
Co-Advisor:	Gamani Karunasiri

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2004	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Infrared Face Recognition	**5. FUNDING NUMBERS**
6. AUTHOR(S) Colin K Lee	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This study continues a previous face recognition investigation using uncooled infrared technology. The database developed in an earlier study is further expanded to include 50 volunteers with 30 facial images from each subject. The automatic image reduction method reduces the pixel size of each image from 160×120 to 60×45. The study reexamines two linear classification methods: the Principal Component Analysis (PCA) and Fisher Linear Discriminant Analysis (LDA). Both PCA and LDA apply eigenvectors and eigenvalues concepts. In addition, the Singular Value Decomposition based Snapshot method is applied to decrease the computational load. The K-fold Cross Validation is applied to estimate classification performances. Results indicate that the best PCA-based method (using all eigenvectors) produces an average classification performance equal to 79.22%. Incorporated with PCA for dimension reduction, the LDA-based method achieves 94.58% accuracy in average classification performance. Additional testing on unfocused images produces no significant impact on the overall classification performance. Overall results again confirm uncooled IR imaging can be used to identify individual subjects in a constrained indoor environment.

14. SUBJECT TERMS Uncooled Infrared Imaging, face recognition, Principle Component Analysis, Fisher Linear Discriminant Analysis, SVD Decomposition, Cross Validation	**15. NUMBER OF PAGES** 155
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

INFRARED FACE RECOGNITION

Colin K. Lee
Lieutenant, the United States Navy
B.S.E.E., Naval Postgraduate School, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2004

Author:	Colin K Lee
Approved by:	Monique P. Fargues Thesis Advisor
	Gamani Karunasiri Co-Advisor
	John P. Powers Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This study continues a previous face recognition investigation using uncooled infrared technology. The database developed in an earlier study is further expanded to include 50 volunteers with 30 facial images from each subject. The automatic image reduction method reduces the pixel size of each image from 160×120 to 60×45. The study re-examines two linear classification methods: the Principal Component Analysis (PCA) and Fisher Linear Discriminant Analysis (LDA). Both PCA and LDA apply eigenvectors and eigenvalues concepts. In addition, the Singular Value Decomposition based Snapshot method is applied to decrease the computational load. The K-fold Cross Validation is applied to estimate classification performances. Results indicate that the best PCA-based method (using all eigenvectors) produces an average classification performance equal to 79.22%. Incorporated with PCA for dimension reduction, the LDA-based method achieves 94.58% accuracy in average classification performance. Additional testing on unfocused images produces no significant impact on the overall classification performance. Overall results again confirm uncooled IR imaging can be used to identify individual subjects in a constrained indoor environment.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author would like to thank Prof. Fargues and Prof. Karunasiri for their guidance and support during the time in performing this study. Their encouragement has earned the author’s utmost respect for their expertise and proficiency.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This study investigated face recognition by using uncooled infrared camera with an expanded database. A database consisted of 420 facial images obtained from 14 volunteers was available from a previous study. An additional 1080 images from 36 volunteers were included in the expanded database, resulting in a total of 1500 images. Each subject was required to perform three different facial expressions with 10 head different orientations. Facial expressions considered were neutral sitting, smiling, and pronouncing the vowel “u”. The distance between the subject and the camera was kept constant while permitting a vertical and horizontal angle freedom of 10°. In addition, 36 were collected with an intentionally unfocused camera lens for additional analysis.

An automatic image cropping technique was developed to accommodate the expanded database. The camera generates 160×120 pixels for each image, and the images size was then reduced to 60×45 pixels by the automatic cropping technique. This study used two linear schemes to investigate infrared imaging for face recognition. The first linear scheme considered was the Principal Component Analysis (PCA). The second linearity approach was the Fisher Linear Discriminant Analysis incorporated with the PCA for dimension reduction and classification. A minimum distance classifier was chosen to determine classification decisions. Different PCA-based and LDA schemes were compared by so-called k-fold cross validation, which used 60% of the images for training and the remaining 40% for testing.

Results indicate that the LDA is far superior to the PCA-based classification algorithm by at least 13% with the expanded database. Overall results show that uncooled infrared imagery can be an efficient and reliable tool for face recognition and classification.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The major advantage of infrared (IR) or thermal imaging is its robustness to illumination changes as it is only subject to emitted radiations from an object. Thermal imaging can detect, identify, and evaluate thermal-related problems in matter of seconds and has been widely used in military applications as described later. It is also a non-invasive diagnostic device, which requires no physical contact with investigated targets. Over the years, IR imaging has been applied to various face recognition applications; however, it still remains unpopular due to its high cost in equipment and maintenance. Recent developments in uncooled IR technology with enhance image resolution and reduce equipment and maintenance costs provided an opportunity to apply this technology for face recognition. This study expands on an earlier study that investigated uncooled infrared imaging in face recognition and classification applications [1].

A. INFRARED TECHNOLOGY BACKGROUND

Sir William Herschel, an astronomer, discovered infrared in 1800 [2]. Knowing that sunlight was made up of all the colors of the spectrum, and that it was also a source of heat, Herschel wanted to find out which color(s) were responsible for heating objects. He devised an experiment using a prism, paperboard, and thermometers with blackened bulbs where he measured the temperatures of the different colors. Herschel observed an increase in temperature as he moved the thermometer from violet to red in the rainbow created by sunlight passing through the prism, and found that the hottest temperature was actually beyond red light. The radiation causing this heating was not visible; Herschel termed this invisible radiation "calorific rays." Nowadays, we called Herschel’s discovery as infrared. Today, infrared technology has tremendous roles in science and engineering.

B. THE INFRARED CAMERA

Infrared radiation is electromagnetic radiation whose wavelengths are greater than those of visible light but shorter than those of microwaves. It is radiated heat invisible to human eye, yet can be sensed by our skin. All objects emit infrared radiation regardless of their temperature. The intensity of the radiated heat is proportional to the fourth power of the absolute temperature of the object. It also depends upon emissivity which is a material property of the object. An ideal infrared emitter, said to be a "blackbody," has an emissivity of unity. Most real objects have emissivities less than unity, and therefore emit less intense infrared radiation than a blackbody at the same temperature does. In summary, temperature and emissivity characterize the infrared emitting properties of an object [3].

Applying the infrared radiation concept, an infrared camera simply detects and converts heat to electrical signal. Resulting electrical signals are then processed to produce a thermal image on a video monitor and to perform temperature calculation [4]. The infrared camera can accurately deliver, identify, and evaluate thermal information. For example, firefighters use thermal imaging to effectively and efficiently locate the origin of the fire and to save many lives. Figure 1 demonstrates an infrared camera's robustness to illumination changes.

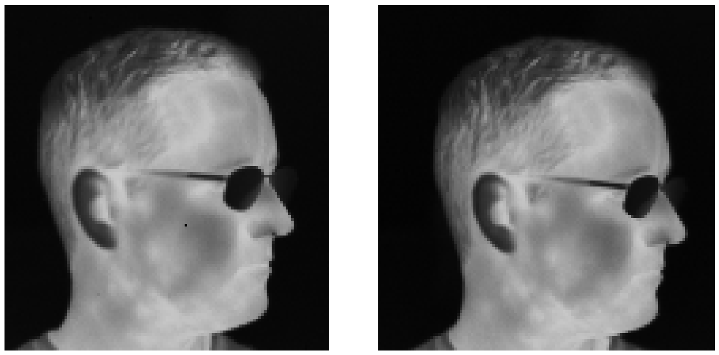

Figure 1. Thermal Image Taken under Complete Darkness (Left) and under Room Light (Right) (From Ref. 9.).

C. THE MEASURE OF TEMPERATURE

Thermal imaging can be used for measuring temperature of an object remotely. For example, infrared cameras with build-in temperature measurement capabilities provide the needed information for electrical and mechanical equipment operating in an optimal condition. Infrared cameras have ability to detect some abnormalities, which are often invisible to naked eyes.

D. INFRARED TECHNOLOGY APPLICATIONS

Infrared technology has a wide variety of applications in both military and civilian industries. Thermal imaging is a critical piece of equipment in today's so called "high tech military as it is totally independent of visible light and can be used in daylight or under complete darkness [5]." Tanks such as the M1A1 Abrams main battle tank have driving systems that rely on thermal infrared imaging to navigate any terrain during day or night. Apache Helicopters have forward-looking infrared (FLIR) units mounted on them that can give the pilot a thermal picture a mile ahead of the aircraft. Fighter planes (F-16, F-14, F-15, F-18, stealth fighter, Tomcat, etc.) use thermal infrared imaging to locate and control fire to specific targets. Even some of the missiles being fired at the Iraqi's are guided with infrared technology (heat seeking missile) [5].

As mentioned earlier, firefighters use thermal imaging effectively to fight fires. It has been proven many times over that firefighters equipped with thermal imagers help save more lives and preserve property. Being able to find trapped victims through dense smoke or darkness is the most obvious application for thermal imaging, but the possible applications are endless. A thermal imager can contribute in a variety of unique ways and become an indispensable firefighting tool [6].

Needless to say, infrared technology has a wide range of applications. Recent development allows affordable thermal imaging for various applications, which have tremendous impact in our society.

E. THE UNCOOLED INFRARED CAMERA

Historically, infrared cameras have used sensors made of materials that require cooling to a temperature equivalent to liquid nitrogen's (77 K) [7]. The cost of making the "cooled" infrared camera is extremely expensive. Through research studies, the production of uncooled high-performance detectors capable of sensing and measuring infrared energy finally has become available. The key technology used in such cameras is the microbolometer focal plane array, originally developed and patented by Honeywell. The microbolometer arrays are fabricated using standard silicon technology coupled with micromachining. Each detector incorporates a monolithic vanadium oxide and silicon microstructure on top of a simplified CMOS read-out integrated circuit. The array used in the present camera (IR-160 manufactured by Infrared Solutions) has 19,200 (160×120) individual detectors, centered on a 51 μm pitch grid. The normal response time of the microbolometer is about 12 ms, enabling operation at 30 Hz. The measured nominal NETD (Noise Equivalent Temperature Difference) is about 50 mK at 30 °C, with a f/0.8 lens. The array structure is tuned for maximum performance in the 8×10^{-6} m to 12×10^{-6} m waveband [7].

With multiple data output options, the IR-160 is extremely flexible and can be integrated into a wide range of applications. The IR-160 is low cost and measures $4.3\times3.9\times4.2$ inches $(W\times H\times D)$ including a 20-mm lens. The IR-160 engine weighs less than 5.0 oz and measures just $3.0\times3.0\times1.5$ inches $(W\times H\times D)$. An 8-bit video image via the RS-232 connection allows for real time data transmission [8]. In addition, the camera has built-in germanium lens, which allows for manual focusing. Further, the camera can be connected to the Hyper Terminal program, which allows users to remotely operate the camera [1].

The purpose of this study is to expand on an earlier study that investigated uncooled infrared imaging in face recognition and classification applications by using 1500 infrared images obtained from 50 volunteers.

This chapter introduced the background information, applications, and current development of infrared imaging. In addition, this chapter explained the reasons for selecting the specific uncooled infrared camera for our study. Chapter II describes the overall system set-up from the camera to the desktop computer. Chapter III presents the image acquisition process, the image files nomenclature, and the automatic image processing. Chapter IV examines the two linear approaches considered, namely the Principal Component Analysis (PCA) and Fisher Linear Discriminant Analysis (LDA) used for face classification. Examples are given to demonstrate the use of both PCA and LDA, as well as compare of the two approaches. In addition, this chapter introduces the minimum distance classifier selected for the classification step. Chapter V describes the k-fold Cross Validation implemented and results obtained. Chapter VI presents the conclusions and recommendations for future study in infrared face recognition. Appendix A includes all MATLAB codes and algorithms implemented in the study. Finally, Appendix B includes all simulation results in spreadsheet format.

THIS PAGE INTENTIONALLY LEFT BLANK

II. THE SYSTEM SET-UP

This chapter describes the overall face recognition system set-up including the infrared camera and interfacing it with the desktop computer. First, we discuss the basic scheme used in this study followed by the software and hardware components, and the choice of equipment.

A. THE BASIC SCHEME

Our overall goal was to collect sufficient data and to use the collected images for training and testing purposes. First, volunteers were required in order to collect thermal images. Next, stored images were cropped and processed for dimension reduction. The processed images retained only the features used for classification. Finally, the database was divided into testing and training sets. Figure 2 depicts the overall face recognition procedure.

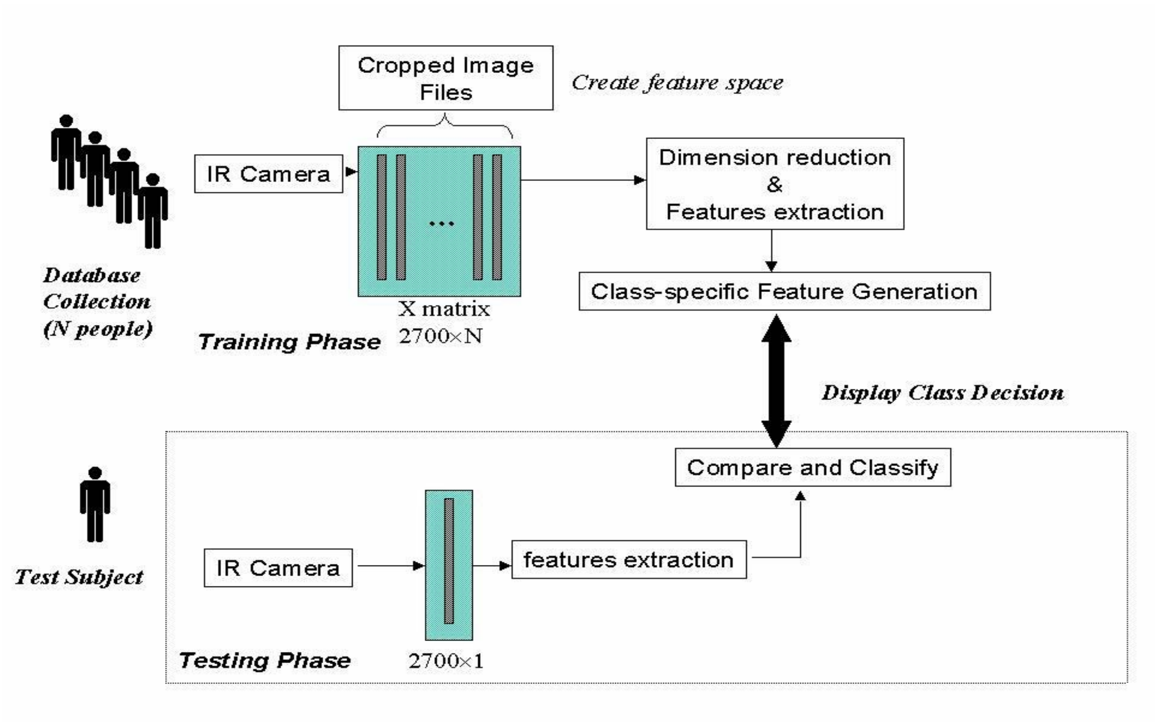

Figure 2. Overall Face Recognition Scheme (From Ref. 9.).

1. Training Stage

The face recognition training stage is illustrated in Figure 3.Training images are used as reference images to perform facial classification. First, all cropped training images are loaded into one data matrix. Next, the projection matrix generated using this data matrix projects the data onto a smaller dimensional feature space. The projected matrix now becomes the reference database and is ready for classification.

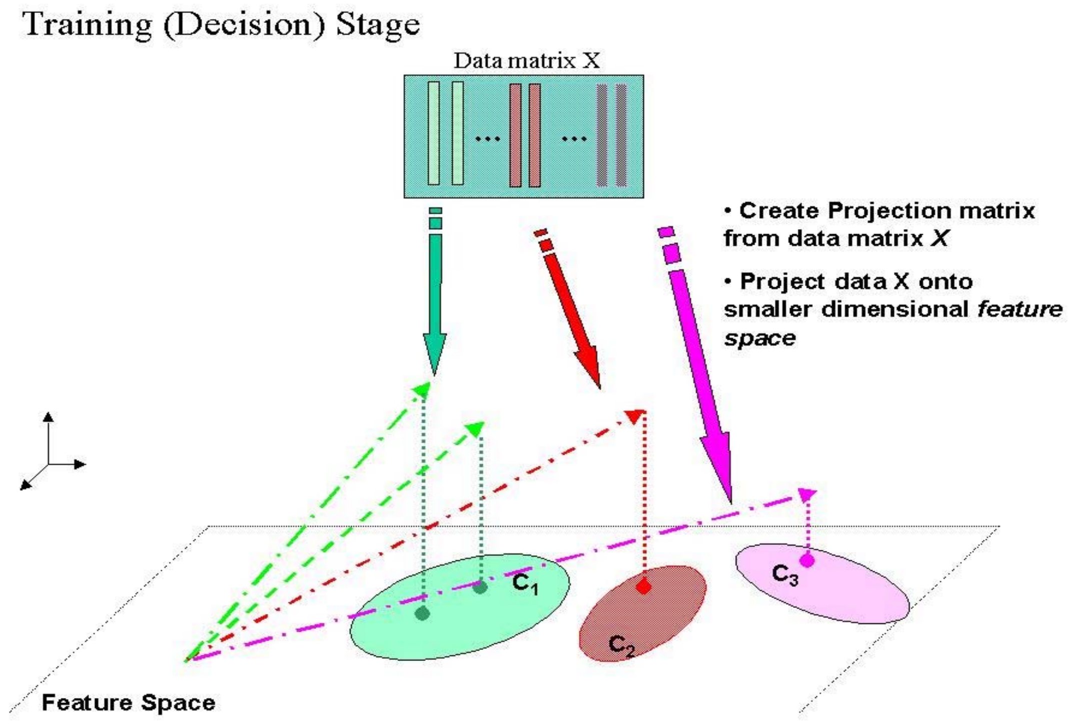
Training (Decision) Stage
Data matrix X
• Create Projection matrix from data matrix *X*
• Project data X onto smaller dimensional *feature space*
C_1
C_2
C_3
Feature Space

Figure 3. Training Stage of Face Recognition (From Ref. 9.).

2. Testing Stage

Figure 4 illustrates the face recognition testing stage. Upon obtaining the training data, testing images are transformed in the same fashion as the training data was. Similarly, testing images are loaded into a single matrix and projected into the smaller dimensional feature space. To classify the "testing" set from the "training" information, we simply look for the smallest distance of the testing data from the centroids of all the training data in the projected feature space.

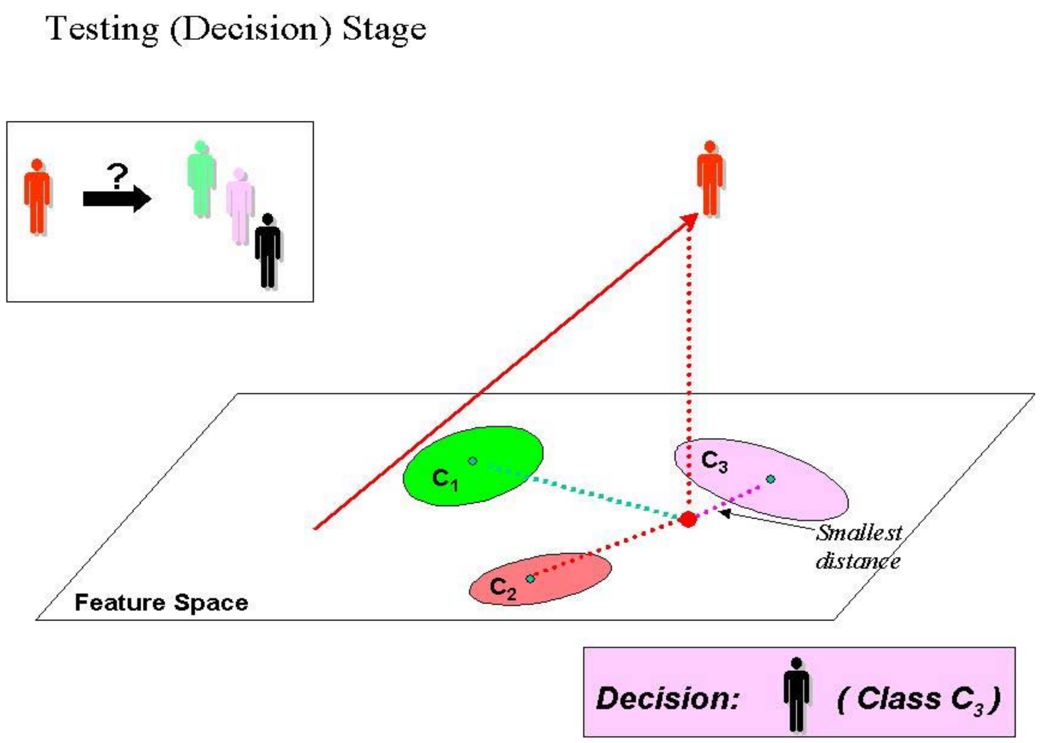

Figure 4. Testing Stage of Face Recognition (From Ref. 9.).

B. SOFTWARE COMPONENTS

As described earlier in Chapter I, the IR-160 uncooled infrared camera from Infrared Solutions Inc. was chosen for the study due its low cost and high performance. All images taken from the cameras are stored in *.pgm* format. Infrared Solutions Inc. provides the software program called "infraview," which can be downloaded from the company website to view the images in *.pgm* format.

Table 1 presents the setting of the HyperTerminal program used in this study. The communication program called HyperTerminal from Windows remotely handles all commands and operations of the IR-160. Users are required to set up the program prior to initiate any operations.

Parameters	**Setting**
Connecting using	COM1
Bits per second	115200
Data Bits	8
Parity	None
Stop Bit	1
Flow Control	Hardware

Table 1. HyperTerminal Program Settings (From Ref. 10.).

Apart from the operations of the infrared camera, MATLAB 6.5 performs all image conversions, processing and storage, and mathematical algorithms.

C. HARDWARE COMPONENTS

Figure 5 displays the hardware components of the entire face recognition system. The entire face recognition system requires an infrared camera for image acquisition, a TV monitor for real-time image display, and a desktop computer for image processing. In order to accurately and efficiently obtain subject images, the ThermaCAM monitor was selected for real-time image display, which allows the user to position the subjects correctly during the data collection process. Similar to the infrared camera, the ThermaCAM monitor connects to the host computer over an RS232 cable. The ThermaCAM is commanded to periodically send out packets of temperature data until told to stop doing so.

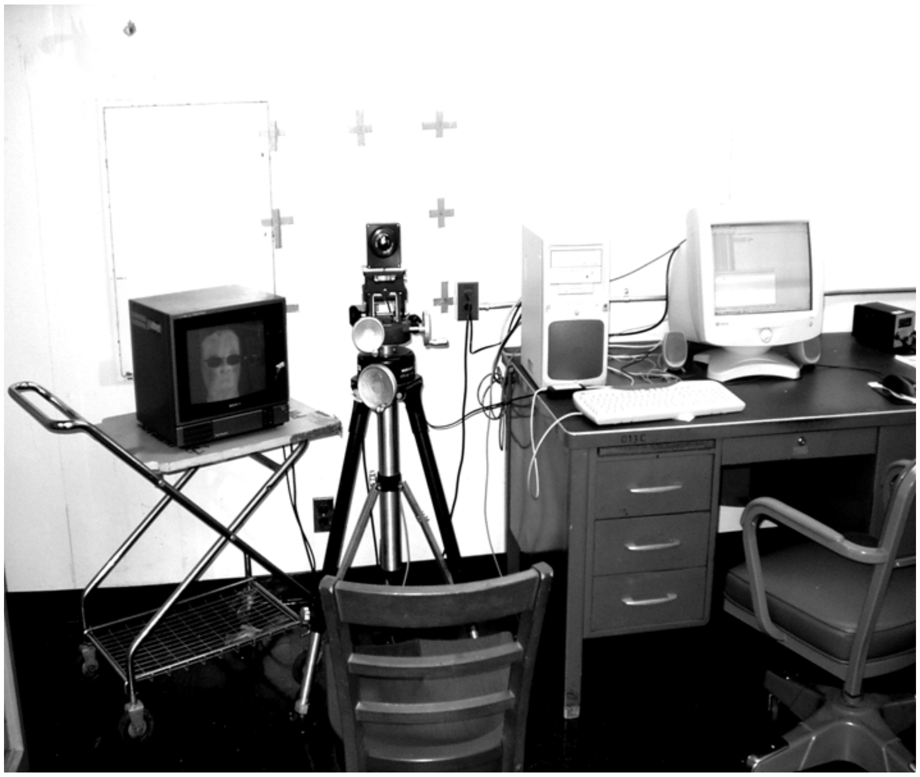
Figure 5. The Hardware Components of Face Recognition System (From Ref. 9.).

This chapter described the overall system set-up used for face recognition. Next, Chapter III presents the data acquisition procedures from photo shooting to file naming.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DATA ACQUISITION

This chapter describes the image acquisition procedures and file nomenclature conventions. In addition, it presents the automatic image cropping schemes and image storing and uploading procedures developed for our study.

A. IMAGE COLLECTION

In order to provide a good set of image data for classification, images must be taken consistently from each individual subject. Hence, it is necessary to have a well-controlled environment to achieve the objective. Parameters under consideration include distances between the infrared camera and subject's face, the height of the camera, and the position of the face. In addition, different facial expressions are required to provide various schemes for examining classification methods. The image acquisition scheme is depicted in Figure 6 and 7.

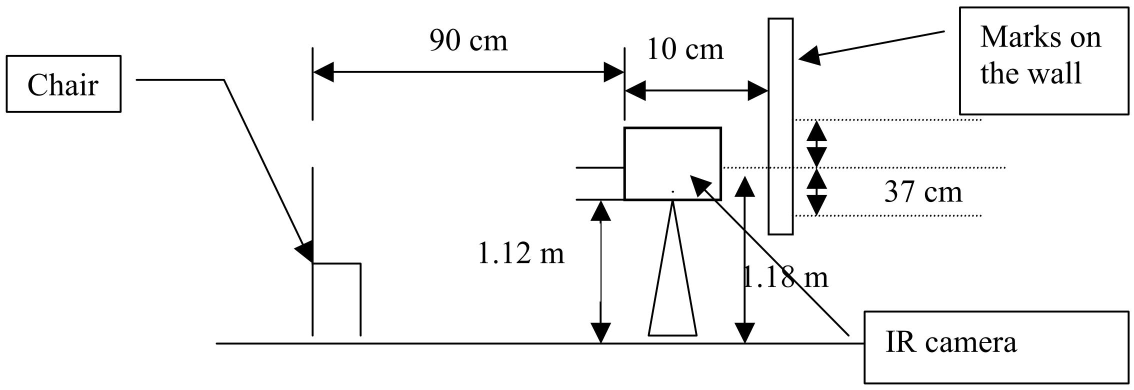

Figure 6. Lateral View Infrared Camera Set-Up (After Ref. 1.).

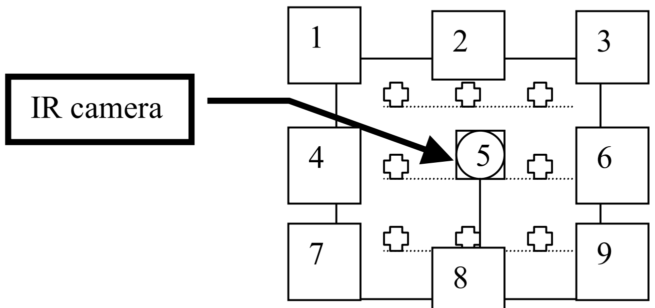

Figure 7. Front View Infrared Camera Set-Up (From Ref. 1.).

Figure 6 is the lateral view of the camera and its distance to the subject. Since the size of the subject's face varies from person to person, it is necessary for the infrared camera to be positioned far enough to cover the entire facial structure but not too far to loose the subject's facial characteristic. Under pre-defined conditions, the distance was set at 90 cm between the camera and the subject's forehead. The height of the center of the camera lens was set at 118 cm from the ground. Next, subjects were required to rotate their head toward 10 different directions to introduce variability in the images collected. Figure 7 depicts the front view of the camera as if the subject looked straight ahead. There were nine numbered points (vary from 1 to 9) marked on the wall. Each subject was asked to turn his or her head toward each number. An additional picture was taken by asking the subjects to look at a random place within the square formed from the extreme marks [1]. Subjects wearing glasses were asked to remove them to limit variations among subjects. Finally, each subject was asked to take an additional picture by looking at the center of the camera lens, which was intentionally unfocused, to investigate the classification scheme robustness to image quality variations. Figure 8 shows a sample of images obtained from the infrared camera.

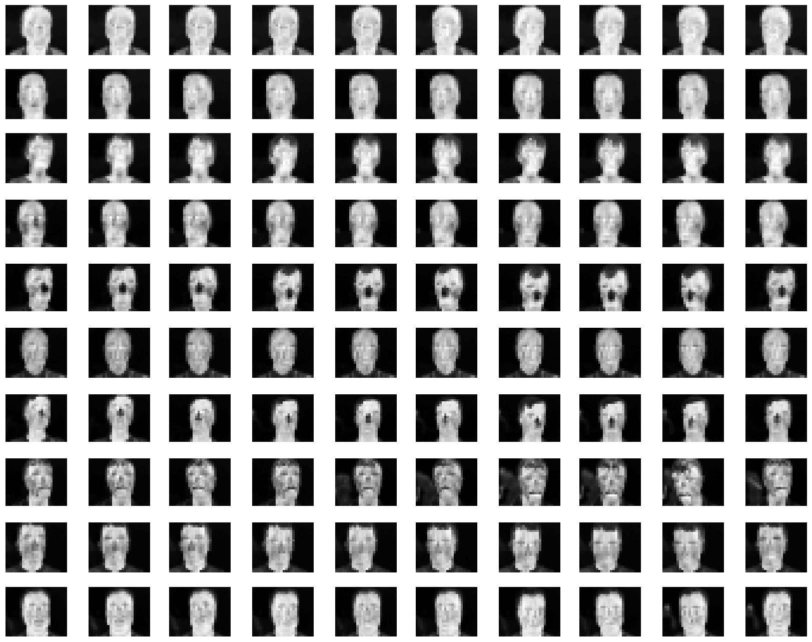

Figure 8. A Sample of Infrared Images (From Ref. 1.).

B. IMAGE NOMENCLATURE

The nomenclature of the images taken from the infrared camera follows that already used in the earlier study [1]: *xx-yy-zz.pgm*, where *xx* is the subject classification number, *yy* is the facial orientation corresponding to the marked number on the wall (Figure 7), *zz* is the facial expression, and *pgm* is the build-in file format from the infrared camera [7]. The range of the selected parameters is the following:

- $[1-6, 8, 9, 11-16, 50-85]$ for *xx*;
- $[1-10]$ for *yy*;
- $[1, 4, 5, 6]$ for *zz*.

The facial expression number *zz* is further defined as follows:

1- a neutral expression;

4- a neutral expression with unfocused lens;

5- a smiling expression;

6- a pronouncing vowel “u” expression.

For example, 51-5-6.pgm represents the subject number 51 with head orientating at marked position number 5 (straight head) while pronouncing “u.” A total of 50 subjects and 31 thermal images each were included in the full database, resulting in a total of 1550 images.

C. IMAGE CROPPING

Infrared images obtained from the IR-160 camera have dimensions equal to 160×120 pixels. To avoid the potential effects due to different backgrounds, all images were cropped to isolate the face-only portions [1]. The resulting cropped images were later used in training and testing stages of the face recognition system. Due to the fact that there were a significant number of images used for the study, an automatic and efficient method of cropping was required to save time and computational costs. Many studies use manual cropping techniques, which require visual inspection and cropping image by image. Others use very complicated automatic methods, which require tremendous computational costs in addition to the face classification algorithm. The method used for this study is very simple and efficient.

The two-step automatic cropping method has the overall objective of reducing the size of the image matrix, while still retaining the useful information for face recognition. First, it eliminates top and bottom image sections which contain background or neck areas and are not useful. Then, it applies similar procedures to crop off the excessive left and right portions. The resulting cropped image only retains the facial characteristics and has dimensions equal to 60×45 pixels. Figure 9 depicts a raw infrared image obtained from one of the subjects.

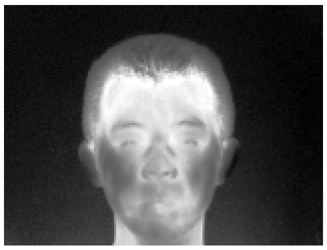

Figure 9. Uncropped Infrared Image with Dimension 160×120 Pixels.

1. “Top and Bottom” Cropping Scheme

Figure 10 summaries the “top and bottom” cropping method. The “top and bottom” automatic cropping scheme retains the facial characteristics below the eyebrows and above the chin. The resulting cropped image reduces the image vertical dimension from 120 to 60 pixels. First, the camera is positioned so that images collected contain only information right above the chin. The MATLAB function “FIND” located the matrix element with elements larger than the threshold value or threshold intensity. In our case, the threshold value is “1,” and the background elements have intensity level less than that. Since MATLAB is column-orientated, the image is rotated 90 degrees counterclockwise before applying the MATLAB function FIND. The FIND function locates the top of the head. The FIND function actually locates the index (ices) of the elements that have intensity greater than the threshold value of “1.” Since the top of the subject’ head is rounded, the FIND function simply returns the index (ices) of very top element(s) it reaches first. The background pixels are then cropped. The image now only has information below the top of the head. Next, we vertically divided the image into half and located the center-line of the image. Taking the center-line of the image as reference, we manually removed all information above the eyebrows. Finally, the image is rotated back to the original vertical orientation. The resulting image now has dimensions of 160×60 pixels. The above cropping method only performs once for one image of each subject and the settings automatically apply to all other images of the same class. The MATLAB function top.m was used to perform top and bottom cropping and is included in the Appendix.

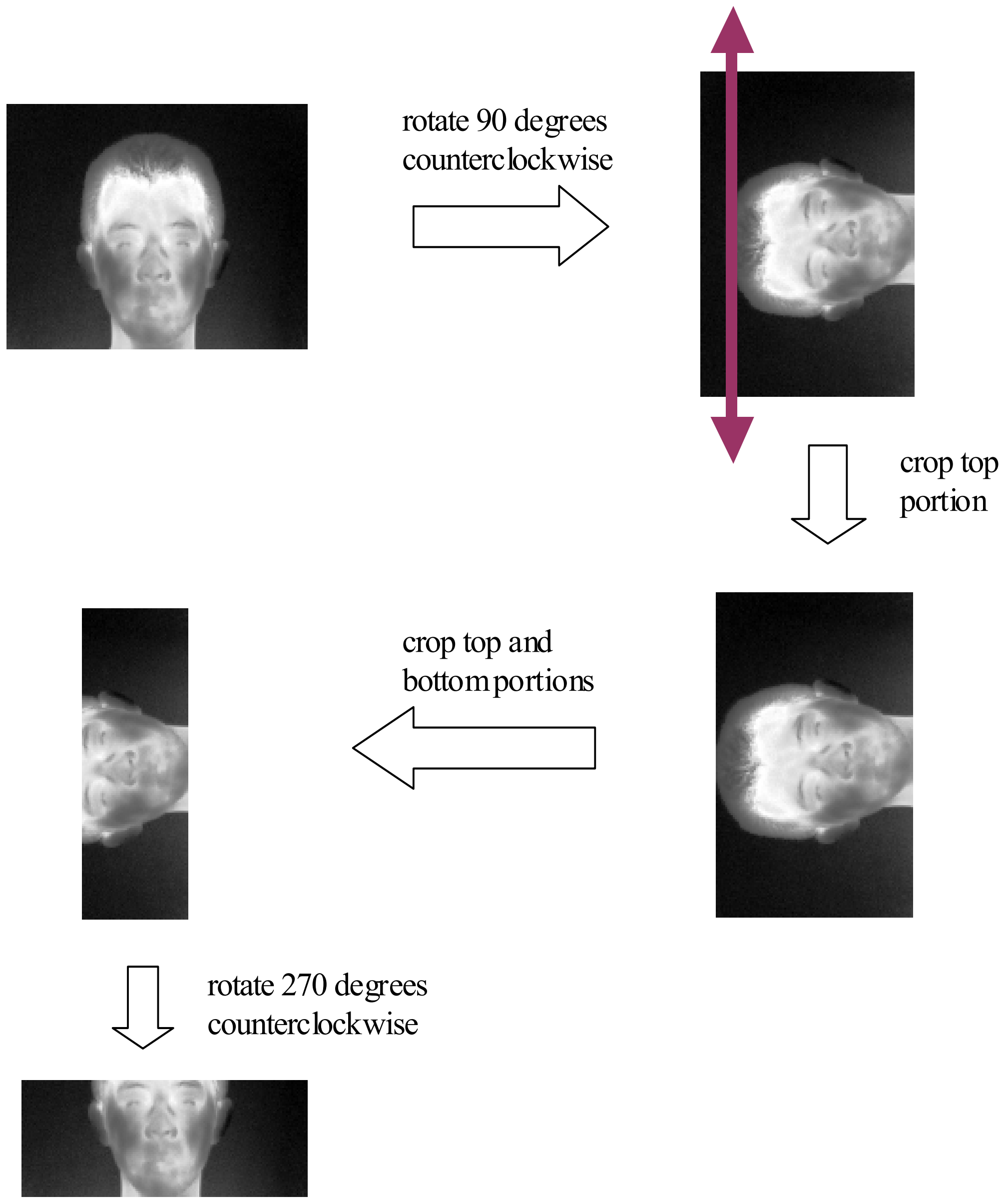
rotate 90 degrees counterclockwise
crop top portion
crop top and bottom portions
rotate 270 degrees counterclockwise

Figure 10. Top and Bottom Cropping Method with Dimension 160×60 Pixels.

2. Side Cropping Scheme

Figure 11 summarizes the side cropping technique, which operates in a similar fashion as the "top and bottom" method. From the "top and bottom" technique, the image has 160×60 pixels. Using the FIND function to locate the leftmost elements of the image matrix, the backgrounds of the left portion are completely eliminated. Then the image is "flipped" to produce the mirror image by using MATLAB function "FLIP." Next, the image is cropped, flipped back to the original orientation, and calibrated to retain only facial elements used for face recognition. The size of the final cropped image is 60×45 pixels.

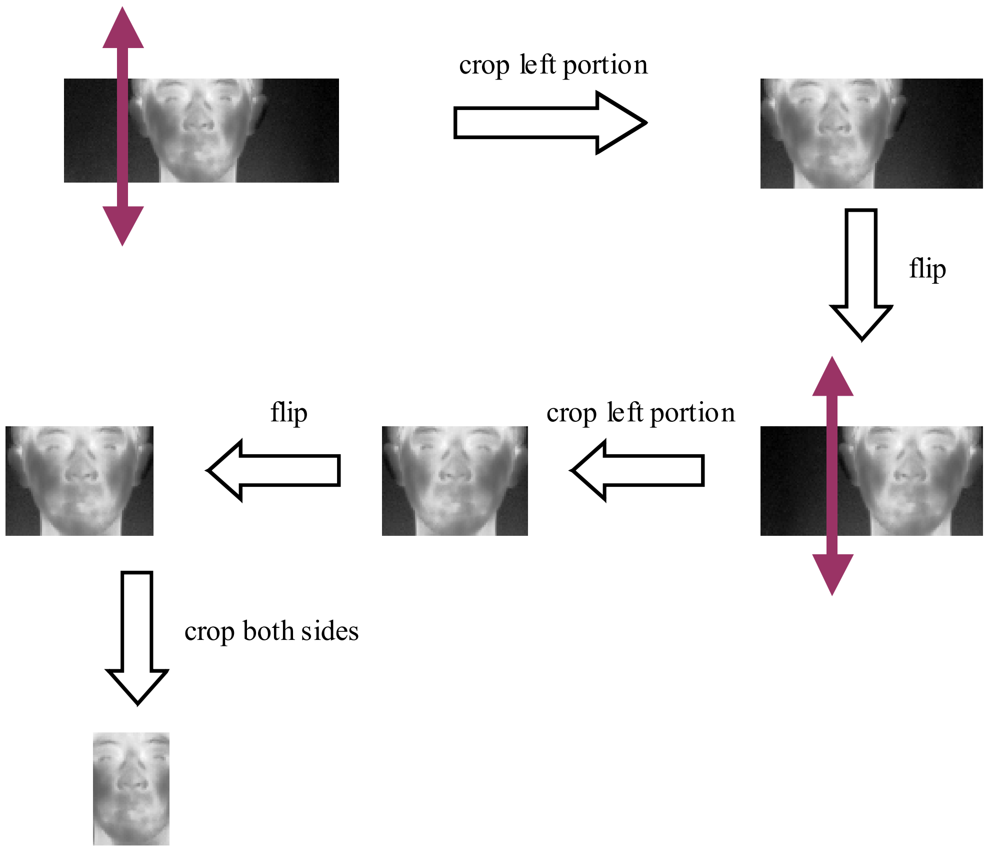

Figure 11. Side Cropping Method with Dimension 60×45 Pixels.

D. DIMENSION REDUCTION (EIGENFACES)

Studies have shown that the Principal Component Analysis (PCA) can be successfully used for data compression. PCA is a linear projection scheme that reduces the dimensionality of a data set while retaining as much variance present in the data as possible [11]. The background information and basic concept is covered in the next chapter. This section describes the use of PCA for dimension reduction.

Since there are total 1550 images in our database, the database size can potentially pose many computational problems for desktop computers or workstations due to the size of the matrices involved in the computations. PCA provides an alternative approach to reduce the dimension of the data matrix and results in lower computational costs. As mentioned above, PCA is merely a linear projection scheme, which aims at eliminating unnecessary or excessive information while keeping the crucial information for data compression purposes. The PCA concept has been applied to face recognition with success both in visible and IR imaging [17, 18]. Images projected into smaller dimensional space are called eigenfaces. Figure 12 and 13 show a sample of the cropped images and eigenfaces obtained with the PCA algorithm. Note that Figure 13 has only 63 eigenfaces vice the original 100 training images due to the dimensionality reduction operation. Figure 14 presents the mean image obtained from all the training images.

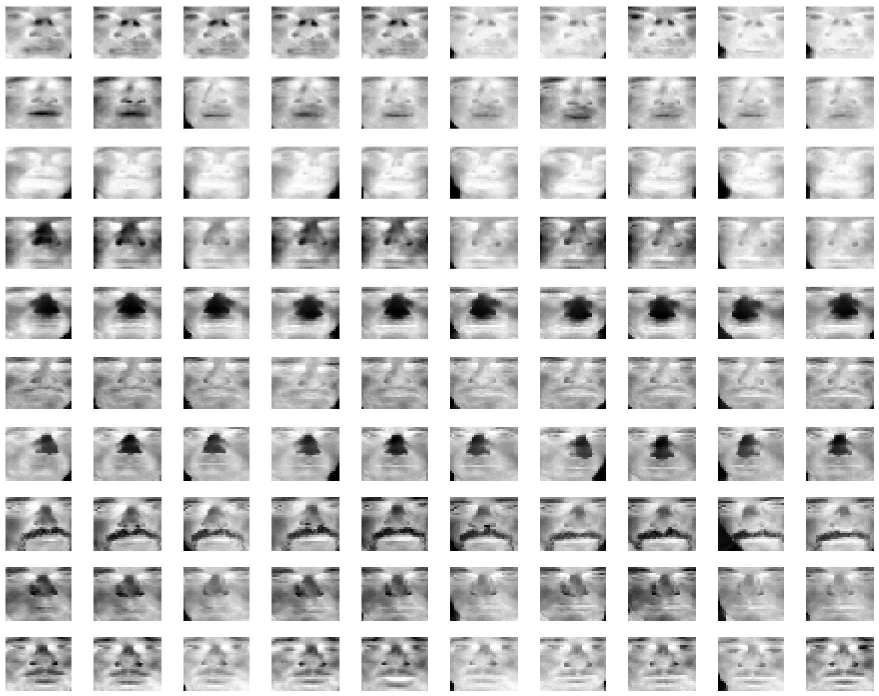

Figure 12. A Sample of Cropped Images.

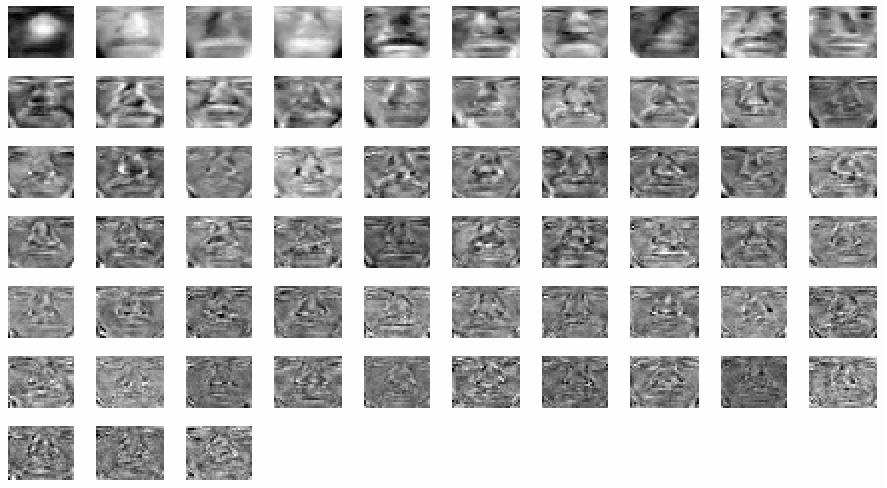

Figure 13. A Sample of Eigenfaces.

Figure 14. Training Data Mean Image.

E. IMAGE STORAGE AND UPLOAD

Overall image collection and storage operations are performed in a systematic fashion. First, raw collected images are stored in the *.pgm* format and assigned class number identifiers, orientation and section numbers according to the nomenclature described earlier in Section B of this chapter. Next, we apply the automatic cropping process and store cropped images in the *.bmp* format. We add the extension "-a" to each cropped image file to distinguish it from the raw image. Finally, all cropped images are reshaped columnwise and stored into a single matrix for algorithm implementation. Since each cropped image has a size equal to 60×45, the associated reshape column vector has a size equal to 2700×1. Hence the resulting data matrix has a size equal to 2700×1500 excluding the unfocused images, which are stored separately. The data matrix is stored in the A_all.mat file. The MATLAB function "load_any_img_to_matrix" allows the user to upload any image into a single matrix for computations. The MATLAB function "load_any_img_to_matrix" file is included in the Appendix.

This chapter covered data acquisition procedures and image file nomenclature conventions. It also described the automatic cropping, dimensionality reduction, and data storage and upload schemes implemented. The next chapter presents the application of the Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) in face recognition.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CLASSIFICATION

This chapter presents the application of the Principal Component Analysis (PCA) and the Fisher Linear Discriminant Analysis, which also incorporates a PCA step, for face recognition. Over the years, both linear methods have widely been used in various face recognition and pattern classification studies with success.

A. PRINCIPAL COMPONENT ANALYSIS (PCA) METHOD

Karl Pearson introduced Principal Component Analysis in 1901 in the analysis of intelligence tests [11]. The basic concept of PCA is to describe the variation of a set of multivariate data in terms of linearly independent (uncorrelated) variables, which are a particular linear combination of the original variables. The new variables are reconstructed in decreasing order of importance. For example, the first principal component measures as much as possible of the variation present in the original data. The overall goal behind the PCA is to determine the least amount of components needed for measuring most of the variation in the data set. These components are then used to represent the original data with little or no loss if information, thus providing a reduction in the dimensionality of the original data and greatly simplifying the analysis [12].

1. Introduction to PCA

Geometrically PCA is a multivariate procedure, which rotates the data such that maximum variabilities are projected onto the axes [13]. Essentially, a set of correlated variables is transformed into a set of uncorrelated variables, which are ordered by reducing variability. The uncorrelated variables are linear combinations of the original variables, and the last of these variables can be removed with minimum loss of real data information.

Let's consider the problem of representing *n*, *k*-dimensional samples of image vectors $\{x_1, \ldots, x_n\}$, where *n* is the number of images (1500), and *k* is the size of each image presented columnwise ($60 \times 45 = 2700$). All images are stacked columnwise in a singe matrix.

The first principal component is the combination of variables that explains the greatest amount of variation. Mathematically, the first principal component y_1 of the observations is the linear combination of the original variables and can be written as:

$$y_1 = a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n, \tag{4.1}$$

whose sample variance is greatest for all coefficients, $a_{11}, a_{12}, \dots, a_{1n}$ (which may be written as the column vector $\boldsymbol{a}_1 = (a_{11}, a_{12}, \dots, a_{1n})^T$). Since the variance of y_1 could be increased without limit, a restriction is placed on these coefficients; as becomes apparent later a sensible constraint is to require that the sum-of-squares of the coefficients, i.e. $\boldsymbol{a}_1{}^T\boldsymbol{a}_1$, should be set to unity.

The second principal component defines the next largest amount of variation and is independent to the first principal component. In other words, y_2 is the linear combination

$$y_2 = a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n, \tag{4.2}$$

i.e., $y_2 = \boldsymbol{a}_2{}^T\boldsymbol{x}$ (where $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$), which has the greatest variance subject to the two conditions: $a_2{}^T a_2 = 1$ and $a_2{}^T a_2 = 0$ (so that y_1 and y_2 are uncorrelated). Similarly the jth principal component is that linear combination

$$y_j = \boldsymbol{a}_j{}^T\boldsymbol{x}, \tag{4.3}$$

which has greatest variance subject to $\boldsymbol{a}_j{}^T\boldsymbol{a}_j = 1$ and $\boldsymbol{a}_j{}^T\boldsymbol{a}_j = 0$ for all $i < j$. To find the coefficients defining the first principal component, the elements of $\boldsymbol{a}_1$ must be chosen to maximize the variance of y_1, subject to constraint $\boldsymbol{a}_2{}^T\boldsymbol{a}_2 = 1$. The variance of y_1 is given by

$$\begin{aligned} \mathrm{Var}(y_1) &= \mathrm{Var}(\boldsymbol{a}_1{}^T\boldsymbol{x}) \\ &= \mathrm{E}\left[\left(\boldsymbol{a}_1^T\left(\boldsymbol{x} - \mathrm{E}(\boldsymbol{x})\right)\right)^2\right]. \end{aligned} \tag{4.4}$$

Since $\boldsymbol{a}^T\left(\boldsymbol{x} - \mathrm{E}(\boldsymbol{x})\right)$ is scalar quantity, the above equation may be rewritten as

$$\begin{aligned}\mathrm{Var}(y_1) &= \mathrm{E}\left[(\boldsymbol{a}_1^T(\boldsymbol{x}-\mathrm{E}(\boldsymbol{x}))(\boldsymbol{x}-\mathrm{E}(\boldsymbol{x}))^T\boldsymbol{a}_1)\right] \\ &= \boldsymbol{a}_1^T\mathrm{E}\left[(\boldsymbol{x}-\mathrm{E}(\boldsymbol{x}))(\boldsymbol{x}-\mathrm{E}(\boldsymbol{x}))^T\boldsymbol{a}_1\right] \quad . \\ &= \boldsymbol{a}_1\boldsymbol{S}\boldsymbol{a}_1.\end{aligned} \tag{4.5}$$

where $\boldsymbol{S} = \mathrm{E}\left[(\boldsymbol{x}-\mathrm{E}(\boldsymbol{x}))(\boldsymbol{x}-\mathrm{E}(\boldsymbol{x}))^T\right]$ is the covariance matrix of the original variables. Note that it is customary to calculate principal components so they have zero mean. Hence, $\boldsymbol{S} = \mathrm{E}\left[\boldsymbol{x}\boldsymbol{x}^T\right]$, becomes the data correlation matrix.

Applying Lagrange multipliers to this maximization problem leads to the solution that $\boldsymbol{a}_1$ is the eigenvector of $\boldsymbol{S}$ corresponding to the largest eigenvalue and, in general, the jth principal component is defined by the eigenvector associated with the jth largest eigenvalue.

If the eigenvalues of $\boldsymbol{S}$ are $\lambda_1, \lambda_{2,} \ldots, \lambda_n$, then, by choosing $\boldsymbol{a}_j^T\boldsymbol{a}_j = 1$, the variance of the jth component is therefore given by λ_j. For example y_1 has variance given by (4.5) and, since $\boldsymbol{a}_1$ is an eigenvector of $\boldsymbol{S}$,

$$\boldsymbol{S}\boldsymbol{a}_1 = \lambda_1\boldsymbol{a}_1. \tag{4.6}$$

So, (4.5) may be written as

$$\begin{aligned}\mathrm{Var}(y_1) &= \boldsymbol{a}_1^T\lambda_1\boldsymbol{a}_1 \\ &= \lambda_1\boldsymbol{a}_1^T\boldsymbol{a}_1 \\ &= \lambda_1,\end{aligned} \tag{4.7}$$

where $\boldsymbol{a}_1^T\boldsymbol{a}_1 = 1$.

Finally, PCA can be viewed as a rotation of the existing axes to new positions in the space defined by the original variables, where there is no correlation between the new variables defined by the rotation. The first new variable contains the maximum amount of variation; the second new variable contains the maximum amount of variation unexplained by the first and orthogonal to the first. The rest of the new variables behave the same way.

2. Snapshot Method

As described earlier in Chapter V, each image contained in the training set is reshaped as a column vector with length *k*, and the data matrix $\boldsymbol{A}$ is defined as the concatenation of all image vector columnwise, resulting in a *k* x *n* matrix, where *k* is the dimension of each image and *n* is the number of training images, respectively [1]. For our study, we use 60% of the data images as training data, which corresponds to a 2700×900 data matrix. Note that the mean image of the training set is subtracted from each image resulting in a matrix $\boldsymbol{X}$. The data correlation matrix $\boldsymbol{S}$ is defined as follows:

$$\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^T . \tag{4.8}$$

The correlation matrix $\boldsymbol{S}$ may have a large dimension, depending on the size of the problem. In our case, we used 60% of the database as the training set, which corresponds to 900 cropped images (each of size 60x45). The resulting $\boldsymbol{X}$ matrix has size equal to 2700×900 resulting in a correlation matrix $\boldsymbol{S}$ of size 2700×2700. As a result, computing the covariance, eigenvectors and eigenvalues can be quite computationally expensive. Note that for a matrix the maximum number of non-zero eigenvectors obtained from $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^T$, where $\boldsymbol{X}$ is of dimension *k* x *n*, is equal to $\min(k,n)$ [15]. Since the number of training images (*n*) is usually smaller than the number of pixels (*k*), the maximum number of non-zero eigenvalues and associated eigenvectors is $n-1$. Recall that the non-zero eigenvalues of $\boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{X}^T\boldsymbol{X}$ are the same [14]. Furthermore, the eigenvectors associated with the non-zero eigenvalues of $\boldsymbol{X}\boldsymbol{X}^T$ are the same as the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$ multiplied by the matrix $\boldsymbol{X}$ and normalized [14]. As a result, the Snapshot method can be used to create the eigenspace from a $n \times n$ matrix rather than a $k \times k$ covariance matrix, as previously shown in [15].

Let *k* and *n* be the image dimension (2700) and the number of training images (900), respectively. The singular value decomposition (SVD) of the $m \times n$ real data matrix $\boldsymbol{X}$ is the factorization

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T , \tag{4.9}$$

where $\boldsymbol{U} \in \boldsymbol{R}^{k \times k}$ is unitary, $\boldsymbol{V} \in \boldsymbol{R}^{n \times n}$ is unitary, and $\boldsymbol{\Sigma} \in \boldsymbol{R}^{k \times n}$ is diagonal. In addition, the diagonal entries σ_j (singular values) of $\boldsymbol{\Sigma}$ are nonnegative and can be stored in non-increasing order; that is, $\sigma_1 > \sigma_2 > \cdots > \sigma_p \geq 0$, where $p = \min(k,n)$ [14]. Recall that the nonzero singular values of the matrix $\boldsymbol{X}$ are the square roots of the nonzero eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$. Note that

$$\boldsymbol{X}^T\boldsymbol{X} = (\boldsymbol{U\Sigma V}^T)^T(\boldsymbol{U\Sigma V}^T) = \boldsymbol{V\Sigma}^T\boldsymbol{U}^T\boldsymbol{U\Sigma V}^T = \boldsymbol{V}(\boldsymbol{\Sigma}^T\boldsymbol{\Sigma})\boldsymbol{V}^T . \tag{4.10}$$

Equation (4.10) indicates that the matrix $\boldsymbol{V}$ contains the eigenvector of $\boldsymbol{X}^T\boldsymbol{X}$ and the eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$ are the diagonal elements of the matrix $\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$ [16]. Further, note that $\boldsymbol{XX}^T$ may be expressed as

$$\boldsymbol{XX}^T = (\boldsymbol{U\Sigma V}^T)(\boldsymbol{V\Sigma}^T\boldsymbol{U}^T) = \boldsymbol{U\Sigma}(\boldsymbol{V}^T\boldsymbol{V})\boldsymbol{\Sigma}^T\boldsymbol{U}^T = \boldsymbol{U\Sigma\Sigma}^T\boldsymbol{U}^T . \tag{4.11}$$

Equation (4.11) indicates that the matrices $\boldsymbol{U}$ and $\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$ contain the eigenvectors and eigenvalues of $\boldsymbol{XX}^T$, respectively. Therefore, both $\boldsymbol{XX}^T$ and $\boldsymbol{X}^T\boldsymbol{X}$ have the same *n* nonzero eigenvalues with $n-p$ additional zero eigenvalues if $n > p$.

Next, consider the product of $\boldsymbol{X}$ and $\boldsymbol{V}$,

$$\boldsymbol{XV} = \boldsymbol{U\Sigma V}^T\boldsymbol{V} = \boldsymbol{U\Sigma} , \tag{4.12}$$

where the last step follows because $\boldsymbol{V}$ is unitary. In addition, the eigenvectors associated with nonzero eigenvalues are given by:

$$\boldsymbol{U} = \boldsymbol{XV\Sigma}^{-1} , \tag{4.13}$$

where $\boldsymbol{\Sigma}^{-1}$ is defined as the inverse of the portion of $\boldsymbol{\Sigma}$ that contains the nonzero diagonal elements of $\boldsymbol{\Sigma}$ only [14].

Therefore, the left eigenvectors associated to non-zero eigenvalues of the potentially very large $n \times n$ covariance matrix $\boldsymbol{XX}^T$ may be computed by computing SVD of the smaller dimensional $k \times k$ covariance matrix $\boldsymbol{X}^T\boldsymbol{X}$.

3. Projectors

The idea to use projection matrices or projectors is to project a set of data into a smaller subspace for data analysis. As mentioned in Chapter III, the projection matrix is created from the data matrix, which is then projected onto the feature space. PCA performs dimensionality reduction by using a projection matrix composed of eigenvectors selected from the eigenvector matrix $\boldsymbol{U}$ (also called "eigenfaces" in face recognition applications) to project the original data into a lower dimensional subspace.

A projector is a square matrix $\boldsymbol{P}$ that satisfies

$$\boldsymbol{P}^2 = \boldsymbol{P}. \tag{4.14}$$

This definition includes both orthogonal and non-orthogonal projectors [14]. Figure 15 illustrates the example for a generic non-orthogonal projection. The term projector might be thought of as rising from the notion that if one were to shrine a light onto the subspace range ($\boldsymbol{P}$) from just the right direction, then $\boldsymbol{Pv}$ would be the shadow projected by the vector $\boldsymbol{v}$. Observe that applying the projector results in $\boldsymbol{v}$ itself if $\boldsymbol{v} \in \text{range}(\boldsymbol{P})$. Mathematically, we have $\boldsymbol{v} = \boldsymbol{Px}$ for some $\boldsymbol{x}$ and

$$\boldsymbol{Pv} = \boldsymbol{P}^2\boldsymbol{x} = \boldsymbol{Px} = \boldsymbol{v}. \tag{4.15}$$

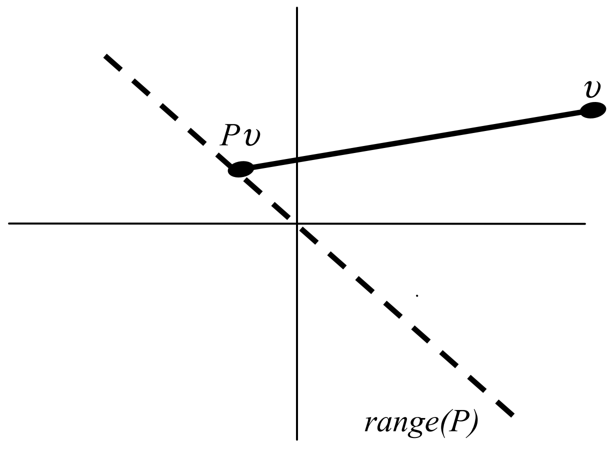

Figure 15. A Non-orthogonal Projection.

Applying the projector to this vector gives a zero result:

$$\boldsymbol{P}(\boldsymbol{P}\boldsymbol{v}-\boldsymbol{v})=\boldsymbol{P}^2\boldsymbol{v}-\boldsymbol{P}\boldsymbol{v}=0. \tag{4.16}$$

This equation means that $\boldsymbol{P}\boldsymbol{v}-\boldsymbol{v}\in \text{null}(\boldsymbol{P})$. That is, the direction of the light may be different for different $\boldsymbol{v}$, but is always described by a vector in $\text{null}(\boldsymbol{P})$.

The orthogonal projector illustrated in Figure 16 is one that projects onto a subspace $\boldsymbol{S}_1$ along $\boldsymbol{S}_2$, where $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are orthogonal. From now on, we mainly deal with the orthogonal projectors.

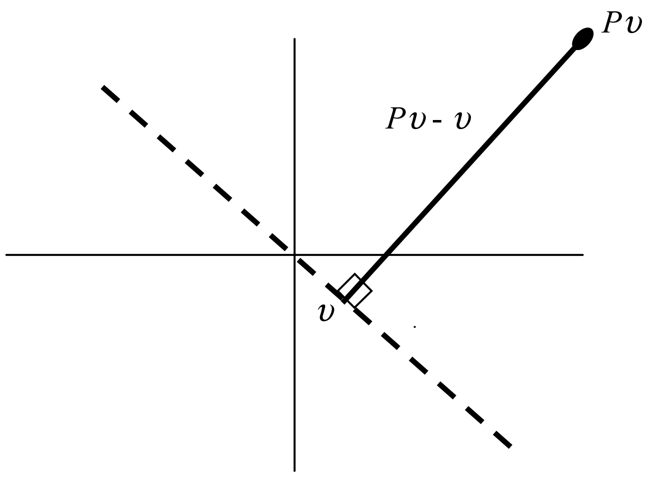

Figure 16. An Orthogonal Projection.

4. An Example of PCA in Dimensionality Reduction

The following example demonstrates the use of PCA in dimensionality reduction [13]. In this example, we take a simple set of two-dimensional data and apply PCA to determine the principal axes. Although the technique can be used with higher dimensional data, two-dimensional data will make it simpler to visualize.

Figure 17 shows a plot of the 100×2 data matrix $\boldsymbol{X}$, which corresponds to a training data containing two images with 10×10 pixels each:

$$\boldsymbol{X} = \begin{pmatrix} -3.072 & -1.7988 \\ -5.6931 & -3.3327 \\ 2.5530 & 1.4945 \\ \vdots & \vdots \\ -0.5495 & -0.3217 \\ 3.0858 & 1.8064 \\ -2.8885 & -1.6909 \end{pmatrix}. \tag{4.17}$$

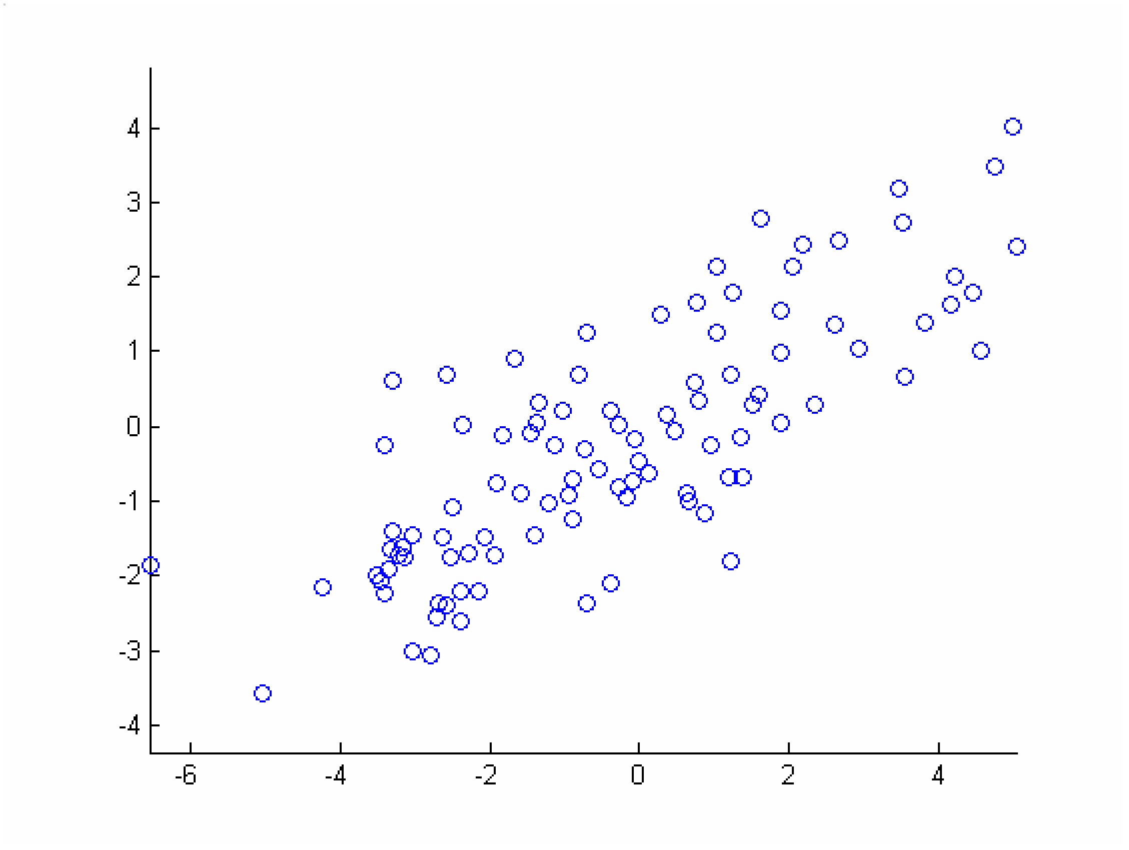

Figure 17. A Scatter Plot of Data Matrix $\boldsymbol{X}$.

Recall from the Snapshot method, the correlation matrix was first computed as follows:

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{pmatrix} 5.9823 & 3.5019 \\ 3.5019 & 2.0500 \end{pmatrix}. \tag{4.18}$$

Next PCA is performed and the principal components were calculated:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} -0.8630 & -0.5052 \\ -0.5052 & 0.8630 \end{pmatrix}. \tag{4.19}$$

Figure 18 shows the first and second principal components axes plotted on top of the scatter matrix $\boldsymbol{X}$. The red and green lines represent the direction of the first and second principal components, respectively. Note how the first principal component lies along the line of greatest variation, and the second lies perpendicular to it. Where there are more than two dimensions, the second component will be perpendicular to the first and along the line of next greatest variation.

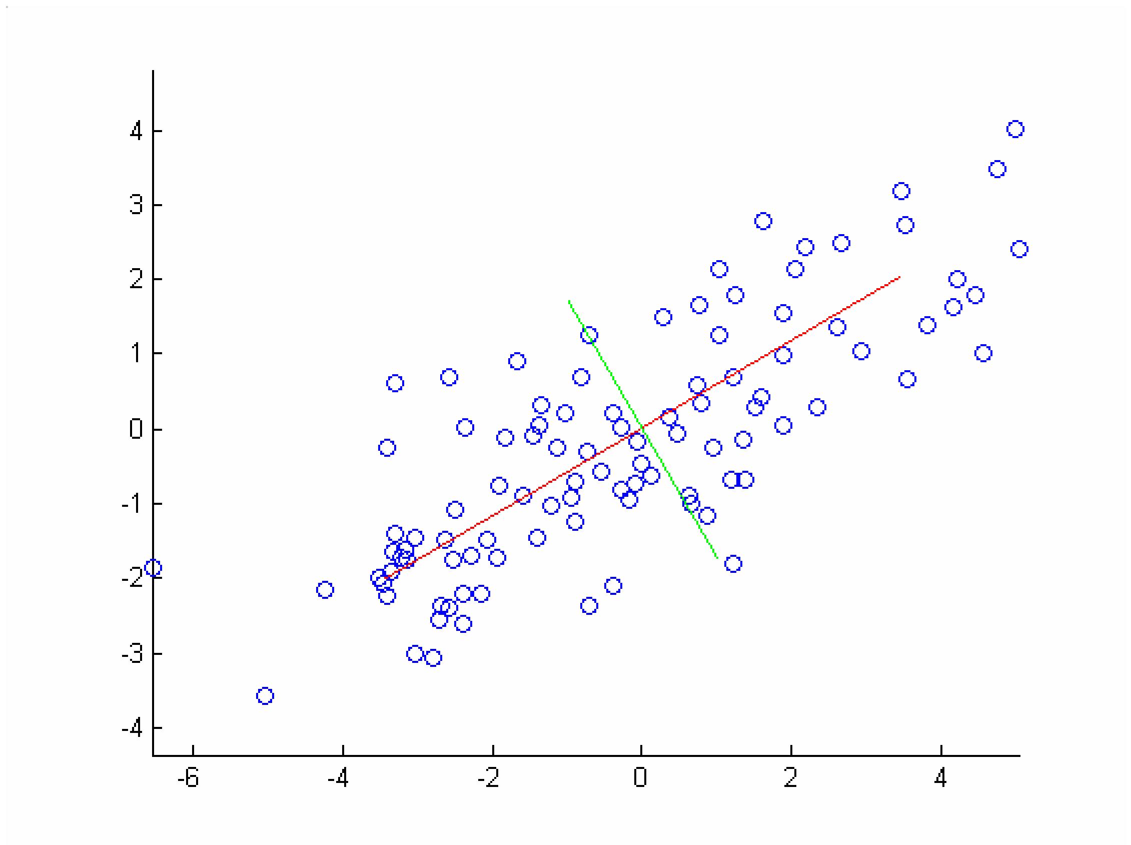

Figure 18. First and Second Principal Components.

Using the projector concept, the original data set was multiplied by the principal components. The data was rotated and laid along the direction of the first principal component. The result is illustrated in Figure 19.

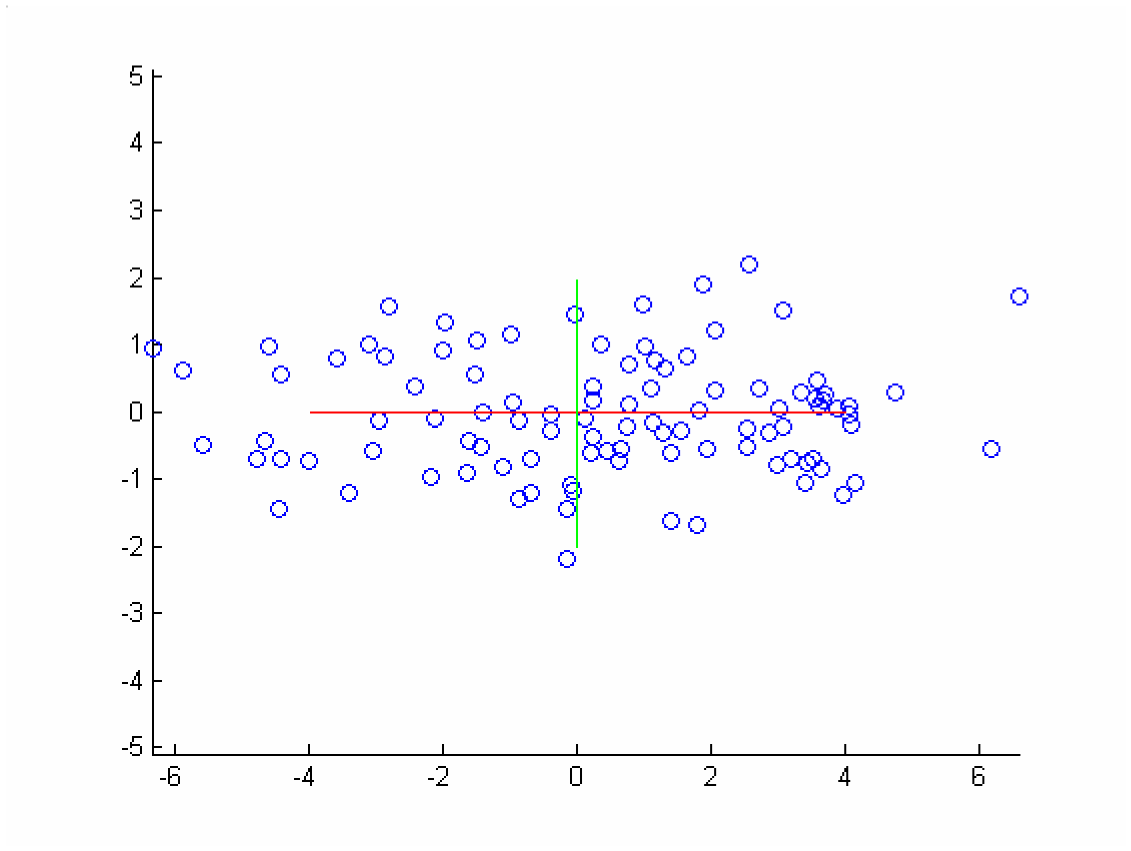

Figure 19. Data lies along with the first Principal Component.

The most common use for the Principal Component Analysis is to reduce the dimensionality of the data while retaining the most information. Figure 20 shows that all the data are projected the direction of the first principal component, thus reducing dimensionality. However, the Principal Component Analysis actually smears the classes together so that they are no longer linearly separable in the projected space [17].

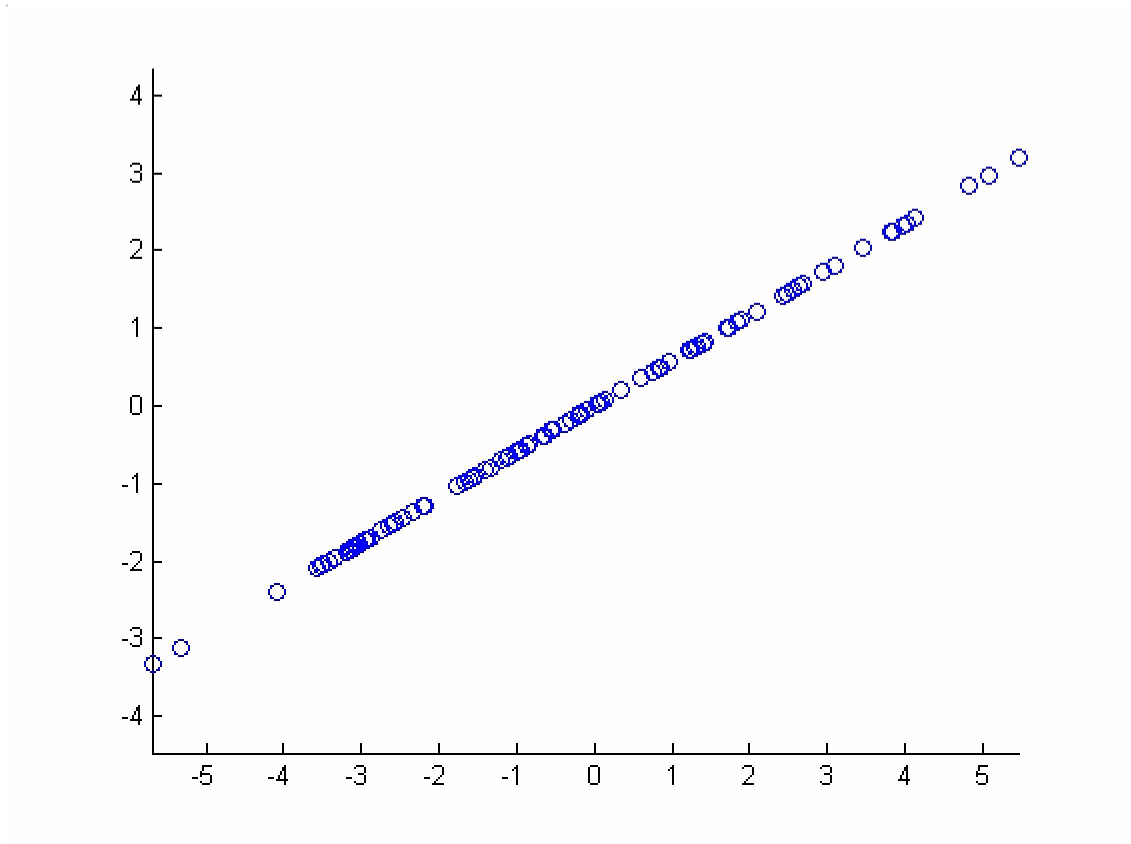

Figure 20. Data Projected onto One Dimension along the First Principal Component.

5. An Example of PCA in Classification

Given two sets of cluster data as shown in Figure 21, we applied the Principal Component Analysis algorithm. In this case each set of clusters is "well" separated from one another which should allow the PCA to classify the two clusters. The green line represents the direction of the first principal component.

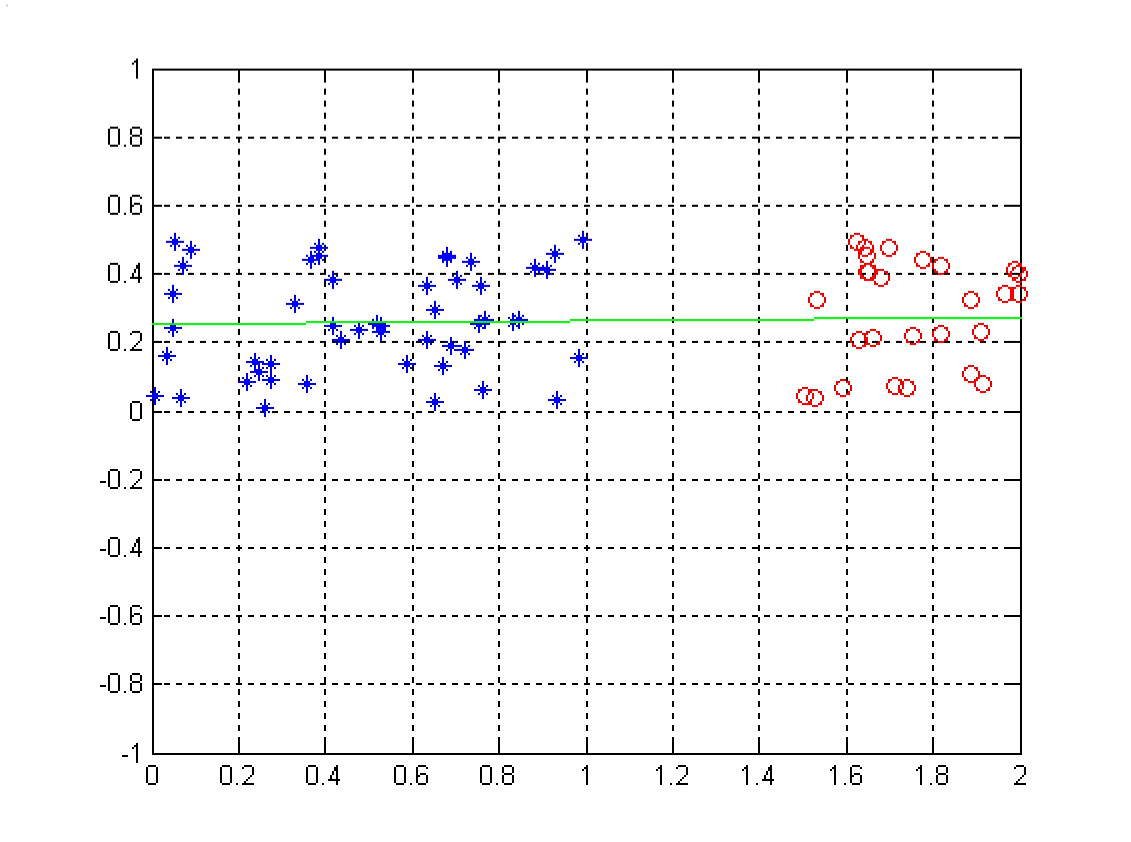

Figure 21. The Principal Component Analysis in Two "Well" Separated Clusters.

The Principal Component Analysis works well for the above example since the clusters are "well" separated and easy to classify.

B. LINEAR DISCRIMINANT ANALYSIS (LDA) METHOD

Today, the most widely used method for face recognition is Fisher's Linear Discriminant Analysis or Linear Discriminant Analysis (LDA). First, we introduce the two-class LDA and next extend to the *C*-class LDA approach.

1. Two-Class LDA

The objective of the Linear Discriminant Analysis is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible [19]. First, we examine the simplest case of the Linear Discriminant Analysis with two classes, and we later generalize to the *C*-class Linear Discriminant Analysis.

Assume we have a set of *D*-dimensional samples $\{x_1, x_{2,} ..., x_n\}$, where *n* is the number of samples, $\boldsymbol{N_1}$ of which belong to class $\boldsymbol{C}_1$, and $\boldsymbol{N_2}$ to class $\boldsymbol{C}_2$. Let's consider a linear transformation mapping the original n-dimensional image space into an m-dimensional feature space, where $m < n$. The new feature vectors $\boldsymbol{y} \in \boldsymbol{R}^m$ are defined by the following linear transformation:

$$\boldsymbol{y} = \boldsymbol{w}^T \boldsymbol{x}, \tag{4.20}$$

where $\boldsymbol{w} \in \boldsymbol{R}^{n \times m}$ is a matrix with orthonormal columns [17]. Of all possible lines we would like to select that which best maximizes the separability of the scalars, as illustrated in Figure 22.

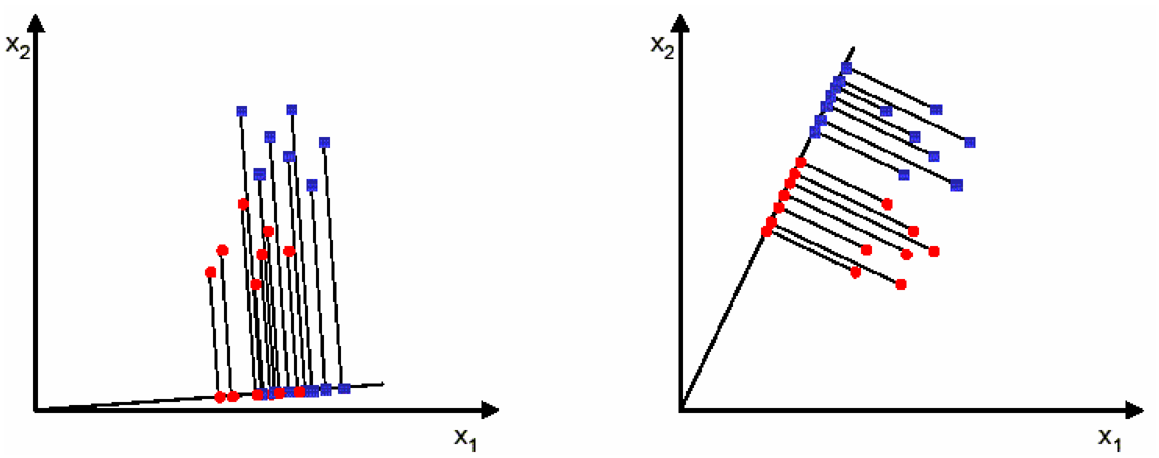

Figure 22. PCA Projection Direction (left graph), LDA Projection Direction (right graph): LDA Projection Maximize Data Separability (From Ref. 20.).

To find a good projection vector $\boldsymbol{w}$, we have to define a measure of separation between the projected classes. The mean vector of each class in $\boldsymbol{x}$ and $\boldsymbol{y}$ feature spaces is defined as

$$\boldsymbol{\mu}_i = \frac{1}{\boldsymbol{N}_i} \sum_{\boldsymbol{x} \in \boldsymbol{C}_i} \boldsymbol{x}, \tag{4.21}$$

and

$$\widetilde{\boldsymbol{\mu}_i} = \frac{1}{\boldsymbol{N}_i} \sum_{\boldsymbol{y} \in \boldsymbol{C}_i} \boldsymbol{y} = \frac{1}{\boldsymbol{N}_i} \sum_{\boldsymbol{x} \in \boldsymbol{C}_i} \boldsymbol{w}^T \boldsymbol{x} = \boldsymbol{w}^T \boldsymbol{\mu}_i. \tag{4.22}$$

We then choose the distance between the projected means as our cost function:

$$J(\boldsymbol{w}) = \left|\widetilde{\boldsymbol{\mu}}_1 - \widetilde{\boldsymbol{\mu}}_2\right| = \left|\boldsymbol{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right|. \tag{4.23}$$

However, the distance between the projected means is not a very good measure since it does not take into account the standard deviation within the classes as illustrated in Figure 23.

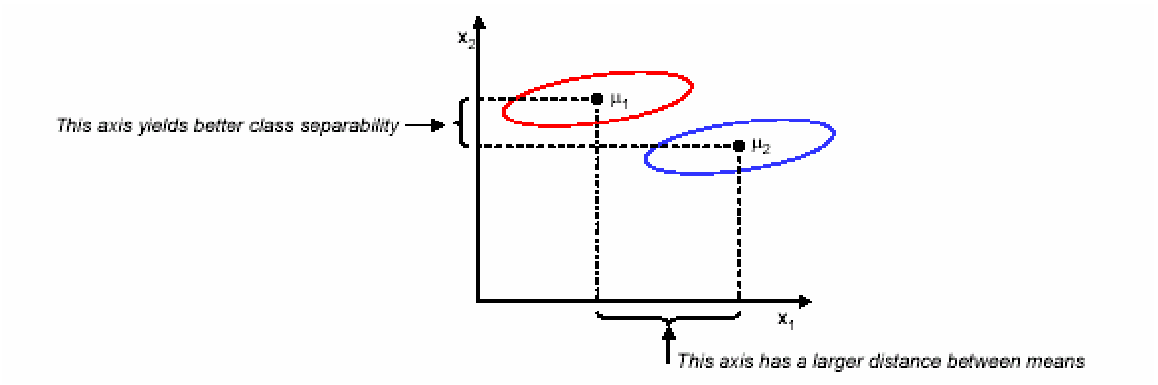
x2
μ1
μ2
This axis yields better class separability
x1
This axis has a larger distance between means

Figure 23. Projected Means onto Cartesian Coordinates (From Ref. 20.).

The solution proposed by Fisher is to maximize a function that represents the distance between the means, normalized by a measure of the within-class scatter. For each class we define the scatter, an equivalent of the variance, as

$$\widetilde{\boldsymbol{S}}_i^{\,2} = \sum_{y \in \boldsymbol{C}_i} (\boldsymbol{y} - \widetilde{\boldsymbol{\mu}}_i)^2 , \tag{4.24}$$

where the quantity $(\widetilde{\boldsymbol{S}}_1^{\,2} + \widetilde{\boldsymbol{S}}_2^{\,2})$ is called the within-class scatter. The Fisher linear discriminant is defined as the linear function $\boldsymbol{w}^T\boldsymbol{x}$ that maximizes the criterion function:

$$J(\boldsymbol{w}) = \frac{\left|\widetilde{\boldsymbol{\mu}}_1 - \widetilde{\boldsymbol{\mu}}_2\right|^2}{\widetilde{\boldsymbol{S}}_1 + \widetilde{\boldsymbol{S}}_2}. \tag{4.25}$$

Therefore, we look for a projection where scatters from the same class are projected very close to each other and, at the same time, the projected means are as farther apart as possible. Figure 24 depicts the projected within class scatters and class means.

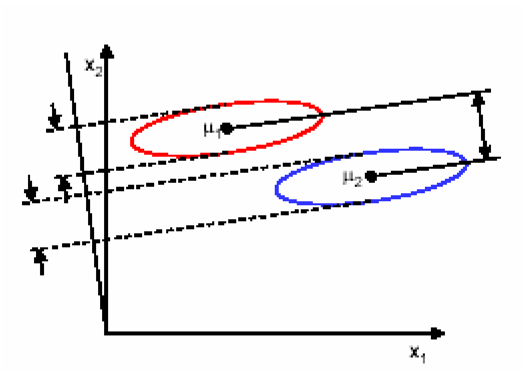

Figure 24. Projected Within Scatters and Class Means (From Ref. 20.).

In order to find the optimum projection w^*, we need to express $J(w)$ as an explicit function of w. We define a measure of the scatter in multivariate feature space x, which are scatter matrices:

$$S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T , \tag{4.26}$$

and

$$S_1 + S_2 = S_W , \tag{4.27}$$

where S_W is called the within-class scatter matrix. The scatter of the projection y can then be expressed as a function of the scatter matrix in feature space x:

$$\widetilde{S}_i = \sum_{y \in C_i} (y - \widetilde{\mu}_i)^2 = \sum_{x \in C_i} (w^T x - w^T \mu_i)^2 = \sum_{x \in C_i} w^T (x - \mu_i)(x - \mu_i)^T w = w^T S_i w , \tag{4.28}$$

and

$$\widetilde{S}_1^2 + \widetilde{S}_1^2 = w^T S_W w . \tag{4.29}$$

Similarly, the difference between projected means can be expressed in term of the means in the original feature space as:

$$(\widetilde{\mu}_1 - \widetilde{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w , \tag{4.30}$$

where the matrix $\boldsymbol{S}_{\mathrm{B}}$ is called the between-class scatter. Note that, since $\boldsymbol{S}_{\mathrm{B}}$ is the outer product of two vectors, its rank is at most one. We can finally express the Fisher criterion in terms of $\boldsymbol{S}_{\mathrm{B}}$ and $\boldsymbol{S}_{\mathrm{w}}$ as:

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}}. \tag{4.31}$$

To find the maximum of $J(\boldsymbol{w})$ we derive and equate to zero:

$$\begin{gathered}
\frac{d}{d\boldsymbol{w}}\left[J(\boldsymbol{w})\right] = \frac{d}{d\boldsymbol{w}}\left[\frac{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}}\right] = 0 \\
\left[\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}\right]\frac{d}{d\boldsymbol{w}}\left[\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w}\right] - \left[\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w}\right]\frac{d}{d\boldsymbol{w}}\left[\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}\right] = 0 \\
\left[\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}\right] 2\boldsymbol{S}_{\mathrm{B}} \boldsymbol{w} - \left[\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w}\right] 2\boldsymbol{S}_{\mathrm{w}} \boldsymbol{w} = 0.
\end{gathered} \tag{4.32}$$

Dividing by $\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}$,

$$\left[\frac{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}}\right] \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w} - \left[\frac{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{b}} \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}}\right] \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w} = 0,$$

$$\Rightarrow \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w} - J \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w} = 0,$$

$$\Rightarrow \boldsymbol{S}_{\mathrm{w}}^{-1} \boldsymbol{S}_{\mathrm{B}} - J\boldsymbol{w} = 0. \tag{4.33}$$

Solving the generalized eigenvalue problem $(\boldsymbol{S}_{\mathrm{w}}^{-1} \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w} = J\boldsymbol{w})$ yields

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}} \left\{\frac{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{B}} \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{w}}\right\} = \boldsymbol{S}_{\mathrm{w}}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \tag{4.34}$$

This above equation is known as Fisher's Linear Discriminant, although it is not a discriminant but rather a specific choice of direction for the projection of the data down to one dimension.

2. *C*-Class LDA

We now generalize Fisher's LDA to the *C*-class solution. Instead of one projection $\boldsymbol{y}$, we seek $(\boldsymbol{C}-1)$ projections $[y_1, y_2, ..., y_{C-1}]$ by means of $(\boldsymbol{C}-1)$ projection vectors $\boldsymbol{w}_p$, which can be arranged by columns into the projection matrix $\boldsymbol{W}=[\boldsymbol{w}_1 \mid \boldsymbol{w}_2 \mid ... \boldsymbol{w}_{C-1}]$:

$$\boldsymbol{y}_i = \boldsymbol{w}_i^T \boldsymbol{x}, \tag{4.35}$$

where $i = 1, 2, \cdots C-1$.

The set of $(\boldsymbol{C}-1)$ projections may be rewritten in a matrix form as:

$$\boldsymbol{y} = [y_1, y_2, ..., y_{C-1}] = \boldsymbol{W}^T \boldsymbol{x}. \tag{4.36}$$

The generalization of the within-class scatter is

$$\boldsymbol{S}_w = \sum_{i=1}^{C} \boldsymbol{S}_i, \tag{4.37}$$

where $\boldsymbol{S}_i = \sum_{\boldsymbol{x}\in \boldsymbol{C}_i} (\boldsymbol{x}-\boldsymbol{\mu}_i)(\boldsymbol{x}-\boldsymbol{\mu}_i)^T$ and $\boldsymbol{\mu}_i = \frac{1}{\boldsymbol{N}_i} \sum_{\boldsymbol{x}\in \boldsymbol{C}_i} \boldsymbol{x}$.

The generalization of the within-class scatter to the C-Class problem is given by

$$\boldsymbol{S}_{\mathrm{B}} = \sum_{i=1}^{\boldsymbol{C}} \boldsymbol{N}_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T, \tag{4.38}$$

where $\boldsymbol{\mu} = \frac{1}{\boldsymbol{N}} \sum_{\forall \boldsymbol{x}} \boldsymbol{x} = \frac{1}{\boldsymbol{N}} \sum_{\boldsymbol{x}\in \boldsymbol{C}_i} \boldsymbol{N}_i \boldsymbol{\mu}_i$.

We also define $\boldsymbol{S}_{\mathrm{T}}$ as the total scatter matrix, which is the sum of $\boldsymbol{S}_{\mathrm{B}}$ and $\boldsymbol{S}_{\mathrm{W}}$:

$$\boldsymbol{S}_{\mathrm{T}} = \boldsymbol{S}_{\mathrm{B}} + \boldsymbol{S}_{\mathrm{W}}. \tag{4.39}$$

Figure 25 illustrates the between-class and within-class scatters.

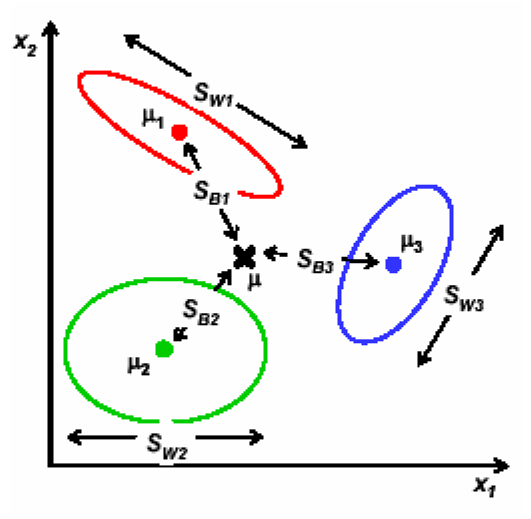

Figure 25. Within-Class ($\boldsymbol{S}_{\mathrm{w}}$) and Between-Class Scatters ($\boldsymbol{S}_{\mathrm{B}}$) (From Ref. 20.).

Recall that we are now looking for a projection that maximizes the ratio of between-class to within-class scatter. Since the projection is no longer a scalar (it has $C-1$ dimensions), we then use the determinant of the scatter matrices to obtain a scalar objective function:

$$J(\boldsymbol{W}) = \frac{\left|\widetilde{\boldsymbol{S}}_{\mathrm{B}}\right|}{\left|\widetilde{\boldsymbol{S}}_{\mathrm{w}}\right|} = \frac{\left|\boldsymbol{W}^T \boldsymbol{S}_{\mathrm{B}} \boldsymbol{W}\right|}{\left|\boldsymbol{W}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{W}\right|}, \tag{4.40}$$

and we seek the projection matrix $\boldsymbol{W}^*$ that maximizes the above ratio.

It can be shown that the optimal projection matrix $\boldsymbol{W}^*$ is the one whose columns are the eigenvectors corresponding to the largest eigenvalues of the following generalized eigenvalue problem,

$$\boldsymbol{W}^* = \left[\boldsymbol{w}_1^* \mid \boldsymbol{w}_2^* \mid \cdots \mid \boldsymbol{w}_{C-1}^*\right] = \operatorname{argmax}\left\{\frac{\left|\boldsymbol{W}^T \boldsymbol{S}_{\mathrm{B}} \boldsymbol{W}\right|}{\left|\boldsymbol{W}^T \boldsymbol{S}_{\mathrm{w}} \boldsymbol{W}\right|}\right\} \tag{4.41}$$

$$\Rightarrow (\boldsymbol{S}_{\mathrm{B}} - \lambda_i \boldsymbol{S}_{\mathrm{W}})\boldsymbol{w}_i^* = 0. \tag{4.42}$$

3. LDA Example

The following example demonstrates the use of LDA in classification.

Given 2 ($=C$) classes, 3 ($=d$) dimensions, and each with 100 ($=n$) samples.

$$\boldsymbol{C}_1 = \begin{pmatrix} -4.33 & -1.67 & \cdots & -0.83 & -0.27 \\ -0.19 & -1.20 & \cdots & 2.19 & -0.12 \\ 6.64 & 5.40 & \cdots & 6.62 & 7.27 \end{pmatrix} \qquad (4.43)$$

and

$$\boldsymbol{C}_2 = \begin{pmatrix} -0.90 & 0.14 & \cdots & -0.19 & -1.05 \\ -0.07 & 0.28 & \cdots & -0.73 & -0.06 \\ -1.44 & 0.61 & \cdots & -1.70 & 0.72 \end{pmatrix}. \qquad (4.44)$$

Figure 26 presents three-dimensional plot of the cluster scatters of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$. Figure 27 shows the perpendicular projection of the data scatters onto two dimensions for visual purpose. Figure 27 shows that it is very difficult to differentiate or separate the two classes when they are projected into the two-dimensional space.

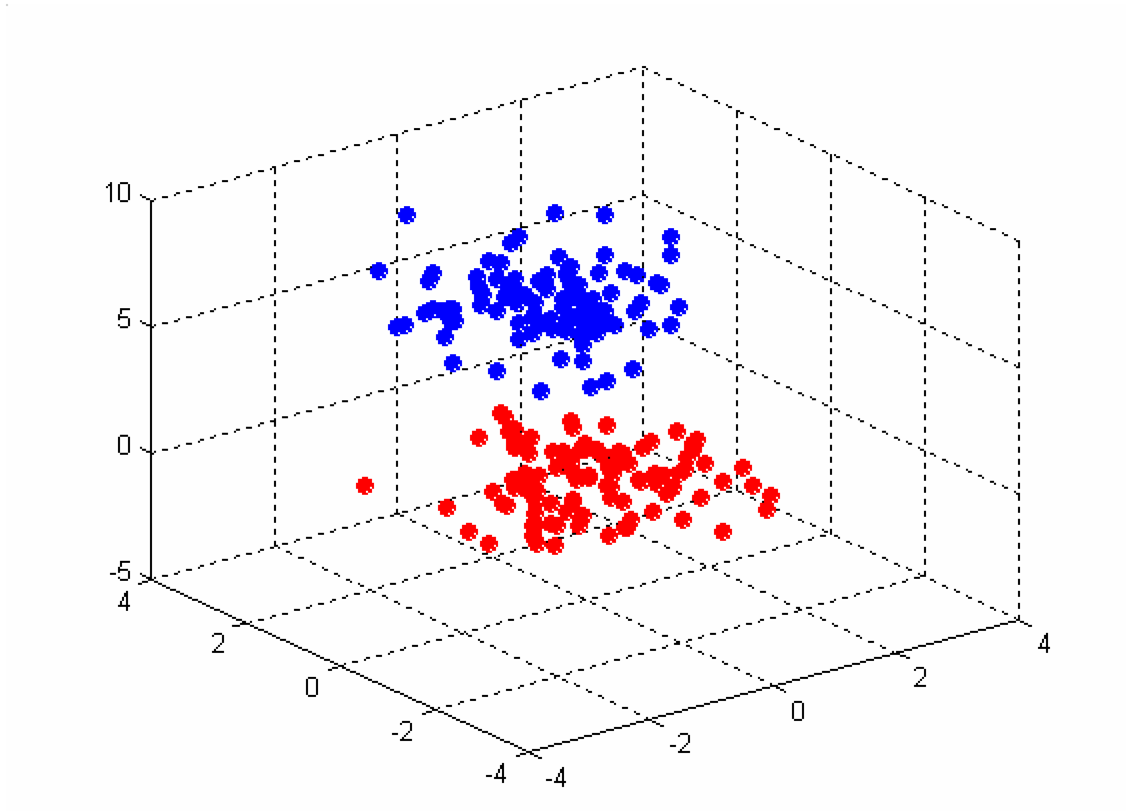

Figure 26. Three Dimensional Cluster Scatters of Class $\boldsymbol{C}_1$ (blue) and $\boldsymbol{C}_2$ (red).

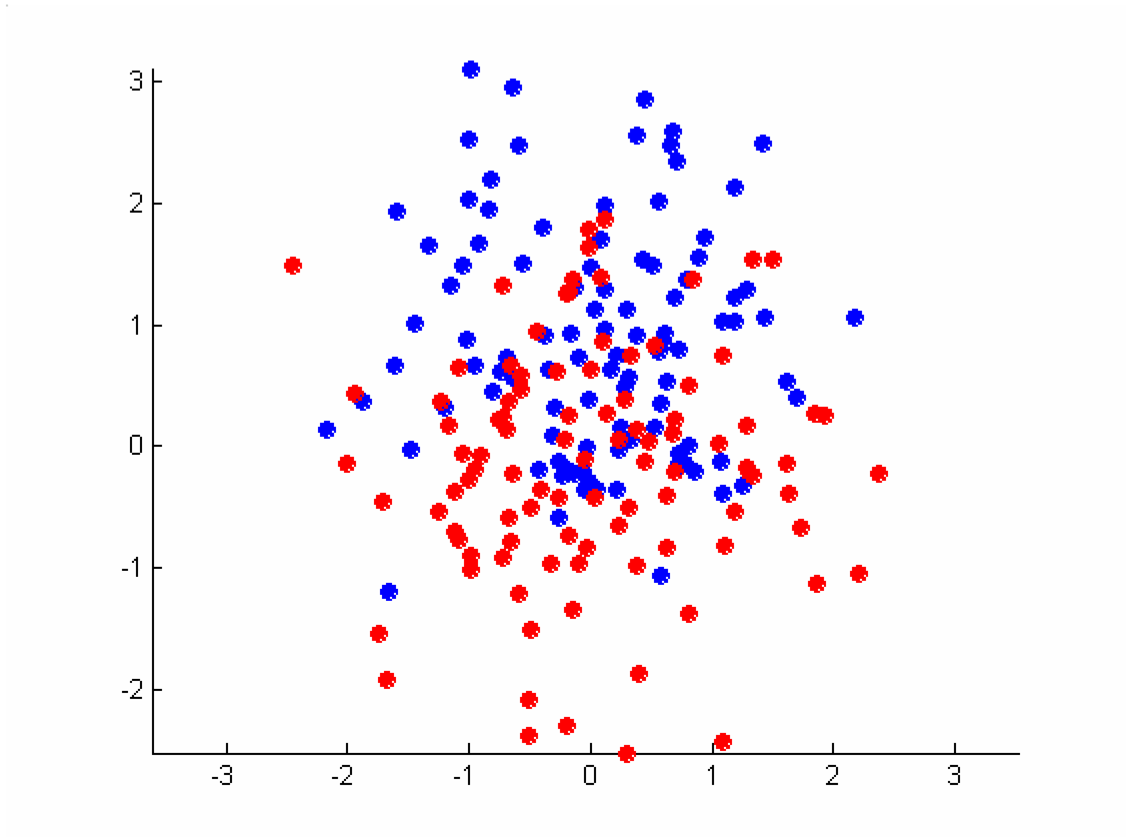

Figure 27. Projected Two-Dimensional Cluster Scatters of Class $\boldsymbol{C}_1$ (blue) and $\boldsymbol{C}_2$ (red).

We now proceed to calculate the within-class $\boldsymbol{S}_{\mathrm{w}}$ and between-class $\boldsymbol{S}_{\mathrm{B}}$ scatter matrices. The results are as follow:

$$\boldsymbol{S}_{\mathrm{W}} = \begin{pmatrix} 371.1504 & 100.2341 & 0.9891 \\ 100.2341 & 65.9105 & 45.9159 \\ 0.9891 & 45.9159 & 228.5554 \end{pmatrix} \qquad (4.45)$$

and

$$\boldsymbol{S}_{\mathrm{B}} = \begin{pmatrix} 1362.3194 & 714.9600 & -468.1968 \\ 714.9600 & 375.2182 & -245.7147 \\ -468.1969 & -245.7148 & 160.9081 \end{pmatrix}. \qquad (4.46)$$

Since we have both $\boldsymbol{S}_{\mathrm{w}}$ and $\boldsymbol{S}_{\mathrm{B}}$, we can seek the projection matrix $\boldsymbol{W}^*$, which is the following:

$$\boldsymbol{W}^* = \begin{pmatrix} -0.95 & -0.06 & -0.74 \\ 0.27 & 0.88 & 0.44 \\ -0.14 & -0.46 & 0.52 \end{pmatrix}. \tag{4.47}$$

To check that $\boldsymbol{W}^*$ maximizes the separability between $\boldsymbol{C}_1$ and $\boldsymbol{C}_1$, we project the original scatters onto $\boldsymbol{W}^*$ in two dimensions. Figure 28 depicts the final results of the Linear Discriminant Analysis of the scatters projected along $\boldsymbol{W}^*$ onto two-dimensional subspace. The green solid line represents the direction of $\boldsymbol{W}^*$ in two-dimensional subspace.

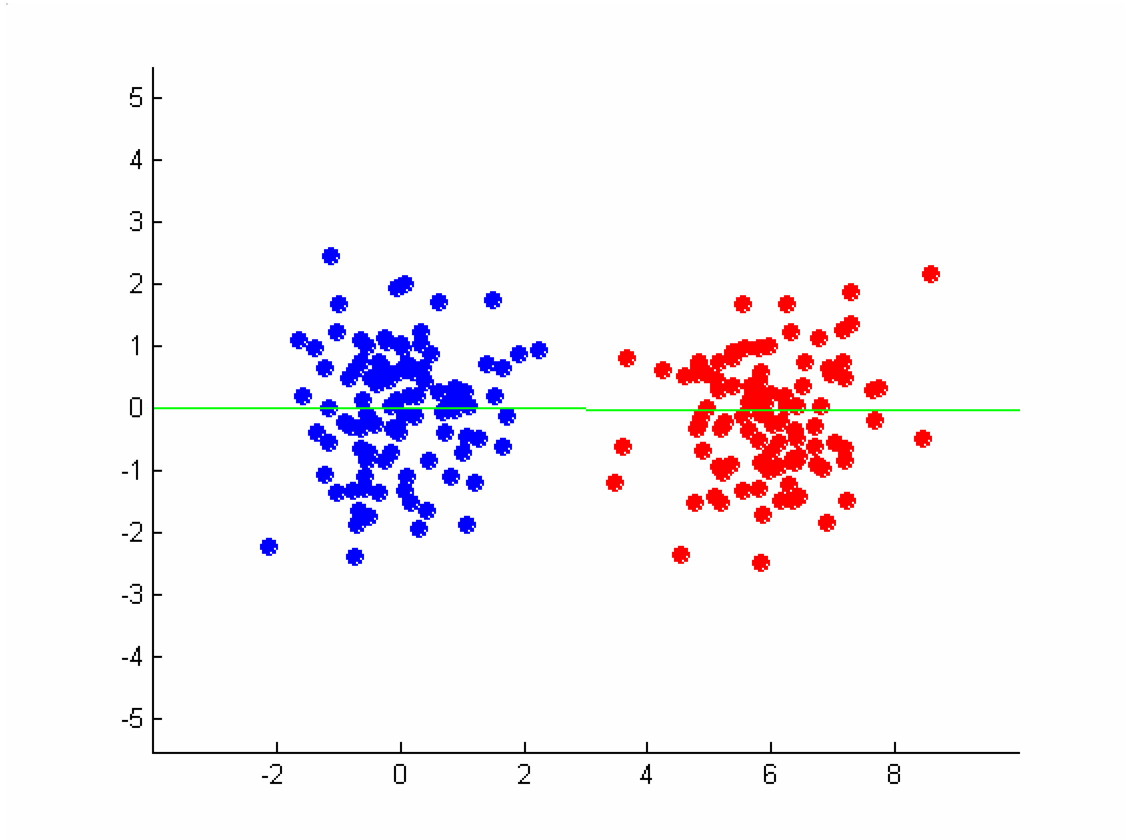

Figure 28. Scatters Projected along $\boldsymbol{W}^*$ in Two-Dimensional Subspace.

C. PCA VERSUS LDA IN CLASSIFICATION

PCA is best used in data compression due to its dimensionality reduction capability [1]. In many cases, the Principal Component Analysis smears the classes together so that they are no longer linearly separable in the projected space [17]. However, LDA is far superior to PCA in pattern classification applications. For example, consider another two sets of clusters, which are oriented in the same directions and lie along the first principal component of the overall data correlation matrix, as illustrated in Figure 29.

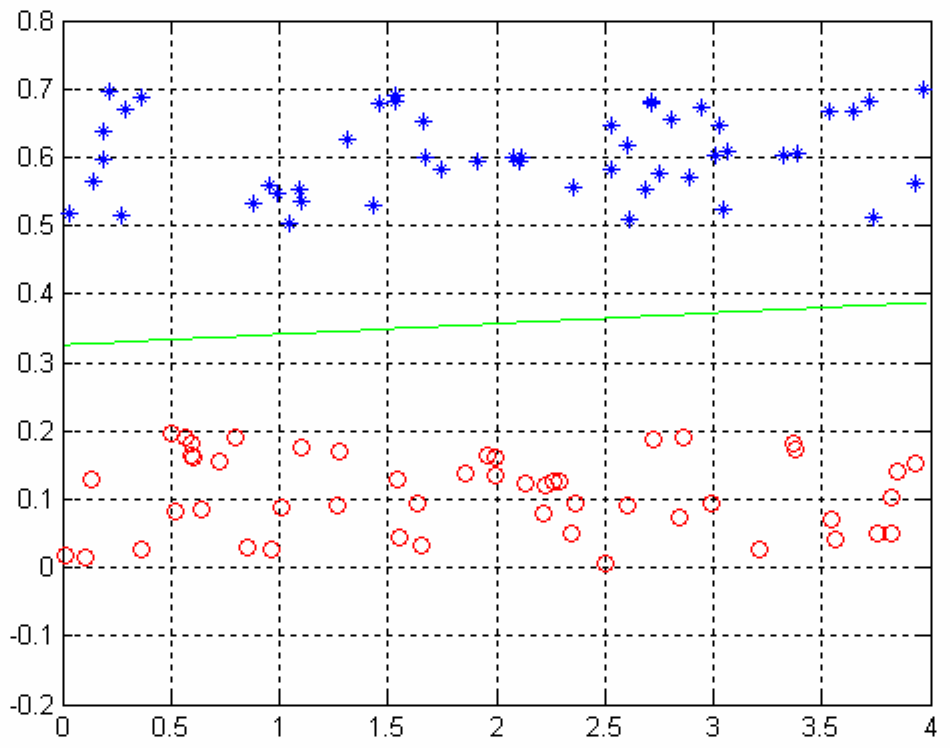

Figure 29. PCA Results on Two "Unseparable" Clusters.

The green solid line in Figure 29 represents the first principal component direction. Note that class discrimination is not preserved when projecting the two classes onto that direction [1].

Now, consider the same set of data given from Figure 29. Figure 30 shows the projection direction obtained with the Linear Discriminant Analysis process as a green solid line. After performing the Linear Discriminant Analysis, the data that was indistinguishable after applying PCA, is now separated by the Linear Discriminant Analysis. The two sets of clusters are no longer oriented in the same directions along the best-projected direction of the LDA, which preserves class discrimination.

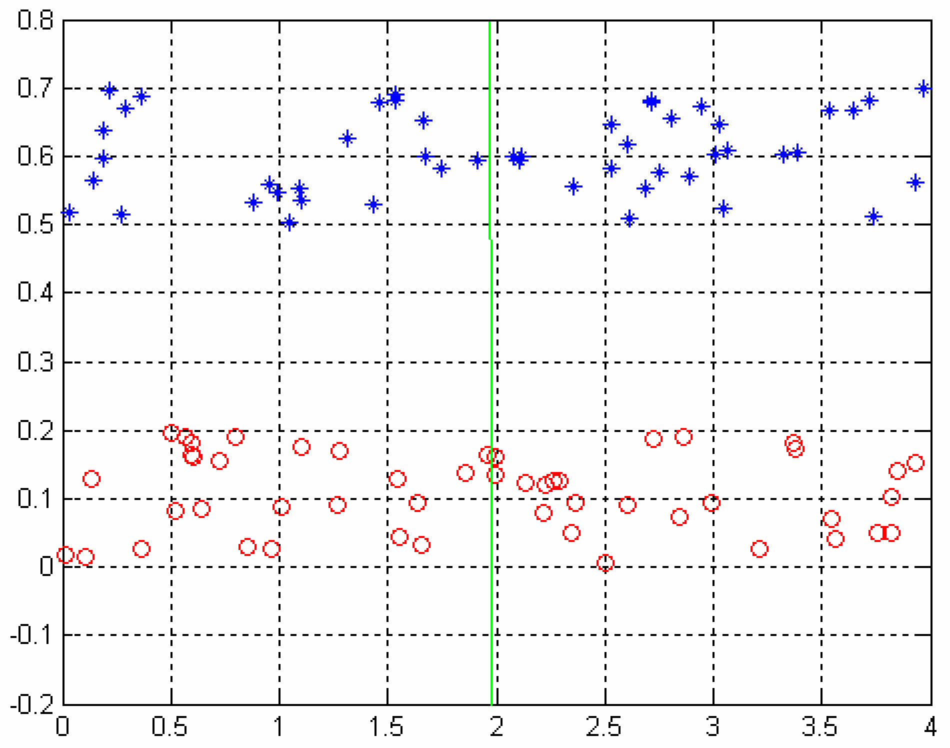

Figure 30. LDA Results on Two-Class Problem.

D. FISHERFACE

Proposed by Belhumeur, Hespanha and Kriegman [17], the method called "Fisherface" is used to avoid the singularity of the within-class scatter matrix $\boldsymbol{S}_{\mathrm{w}}$ by first projecting the image data onto a lower dimensional subspace to produce a non-singular matrix $\boldsymbol{S}_{\mathrm{w}}$. PCA is applied to reduce the dimension from n to $n-C$, where n is the total number of images in the data set and C is the total number of classes. Next, LDA is applied to further reduce the dimension to $C-1$. This approach was successful with the smaller database [1], therefore we also applied the Fisherface approach to the expanded database in our follow-on study.

E. CLASSIFIER

The database is divided into two non overlapping sets; the training set contains 60% of the data per class and the testing set contains the remaining 40%. Next we used PCA and LDA to reduce the dataset dimensions. The overall objective is to assign each image from the testing set to a corresponding class. First, 50 class-specific centroids are extracted from the training data are computed to represent each class. The class-specific centroids are obtained by computing the average values of the projected training data for each class and used to represent each class [1]. Next, the classification decision is made by computing the distance between projected testing image data features and class centroids, and selecting as class that leading to the smallest distance.

This chapter presented the basic concepts behind the Principal Component Analysis and the Fisher Linear Discriminant Analysis. We described projection operations applied to extract class-specific information from the training dataset, and the class decision process. In addition, examples were presented in this chapter illustrate the differences between PCA and LDA. The next chapter presents the overall experimental results obtained with our database.

V. RESULTS

To determine the performance of PCA and LDA-based methods used for face recognition, we implemented k-fold cross validation. K-fold cross validation is a statistical scheme that can be applied to estimate the generalization error of a given model, or to select one of several potential models with the smallest estimated generalization error [21]. In our case, we want to select the algorithm among various PCA-based schemes and the LDA implementation with best classification performances. K-fold cross validation is quite successful for small databases. The 1500 image database, corresponding to 50 subjects with 30 images each was fully utilized. In addition, this chapter presents unfocused infrared camera lens impacts on classification performances.

A. K-FOLD CROSS VALIDATION

Cross-validation is a method designed for estimating the generalization error based on "resampling" [21]. In k-fold cross-validation, the data set is divided into *k* subsets and trained *k* times, each time leaving out one of the subsets from training, but using only the omitted subset to compute error criterion. For each *k* experiment, we used $k-1$ folds for training and the remaining for testing. The resulting error is estimated as the mean error rate [22] and defined as:

$$E = \frac{1}{k}\sum_{i=1}^{k} e_i , \tag{4.48}$$

where, e_i is error rate of each k experiment. Figure 31 depicts the concept behind k-fold cross validation.

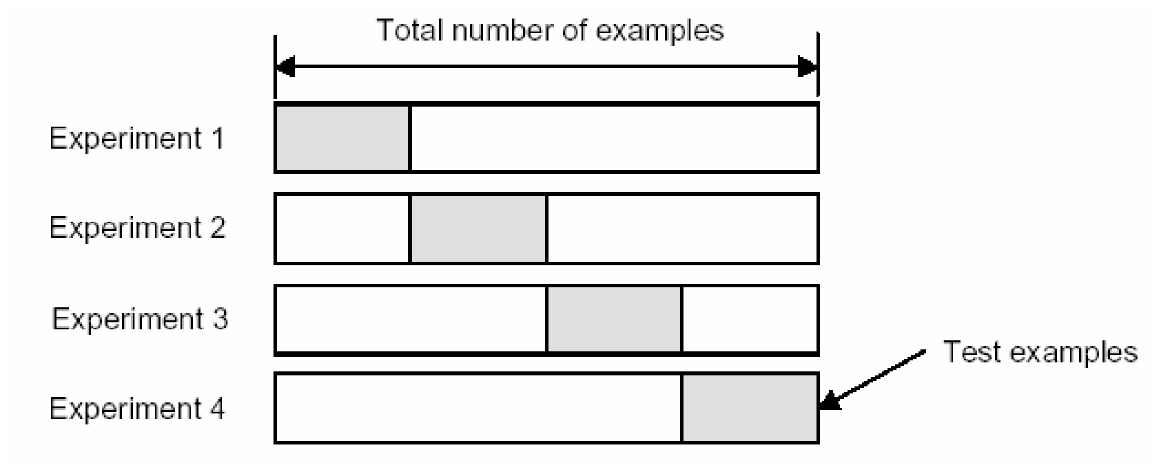

Figure 31. K-Fold Cross Validation.

Figure 32 illustrates the use of k-fold cross validation in our study. For each experiment, the database is split into non-overlapping testing and training sets. The training set includes 60% of each class data and is used to compute projection directions and class centroids. The testing set contains the remaining 40% of the images and is used to test the overall classification performance. The process is trained *k* times that correspond to the number of training subsets. Each time one of the subsets from the training set is omitted and only this omitted subset is used to compute the mean error. To ensure the k-fold cross validation is suitable for our experiment, we tested different *k* values (100, 200, 500, 900, and 1000). Results indicated any values above 200 produce less than 1% deviation in resulting overall classification performances. Therefore, we chose $k = 900$ our study for the general results. The overall classification performance corresponds to the mean and median error rates obtained from all experiments.

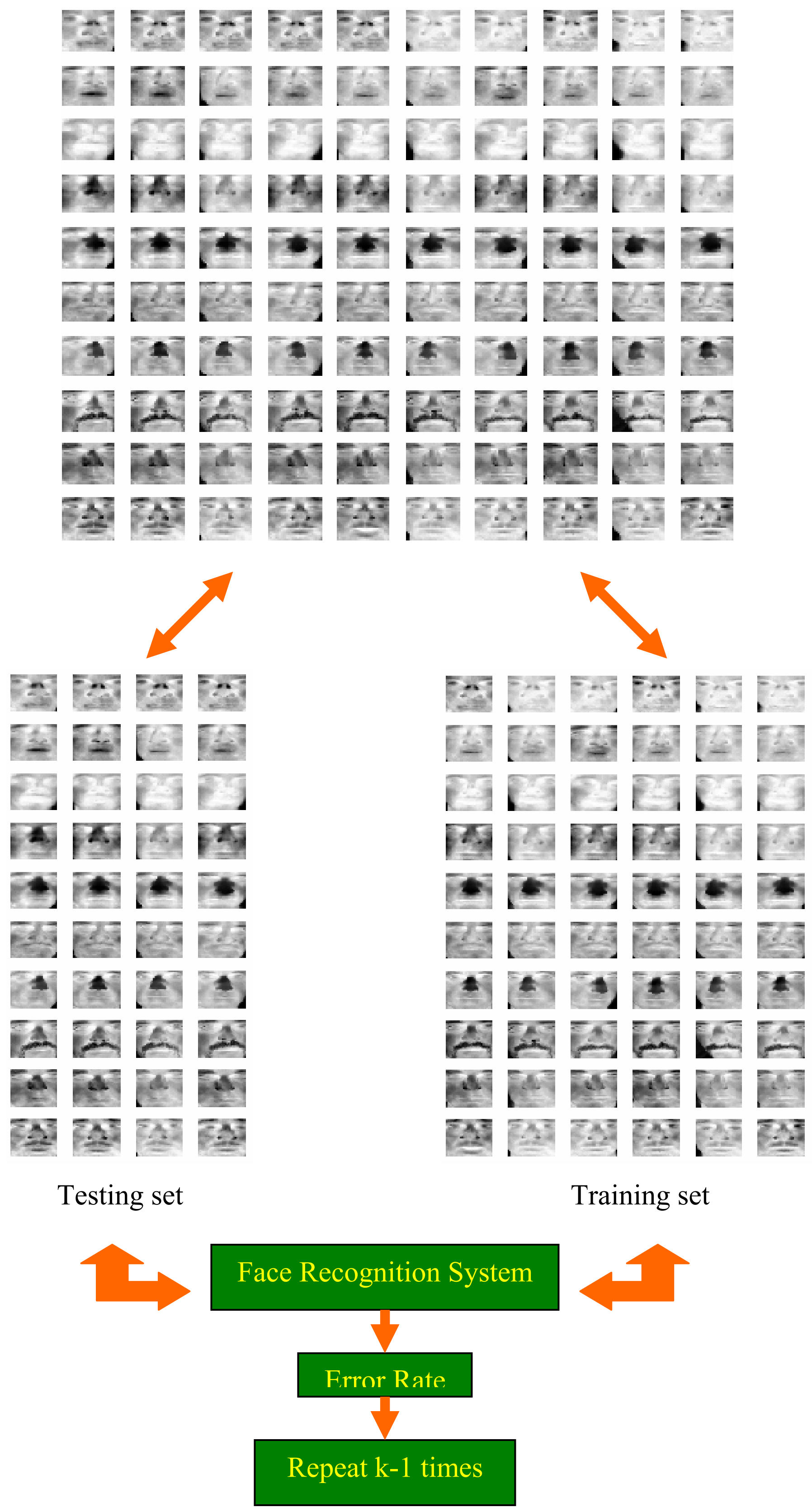

Figure 32. K-fold Cross Validation in Face Recognition.

B. CHOOSING EIGENVECTORS

One of the objectives of our study is to design an effective and efficient model that is best suited for IR face recognition. We initially selected the Principal Component Analysis for its superiority in dimensionality reduction. First, we define the number of "useful" eigenvectors obtained from the PCA, as those associated to eigenvalues equal to at least equal 0.1% of the maximum eigenvalue, following the earlier study by Pereira [1]. By simply removing eigenvectors with associated eigenvalues below the user-specified threshold, we reduce the problem dimension and lower the computational cost without degrading the classification performance [17].

Figure 33 presents the error rate as a function of the number of eigenvectors used for the PCA-based schemes. The maximum number of PCA-based projections is directly related to the size of the data used. Some eigenvectors and the corresponding projection directions can be eliminated because the associated eigenvalues are close to or equal to zeros. In reality, the maximum number of PCA-based projection directions associated with non-zero eigenvalues is equal to the size of the dataset when the database size is smaller than the data dimension [1]. In this study, the maximum number of PCA-based projections directions associated with non-zero eigenvalues is presumably 900 (60% of 1500 images), which is far less than the maximum number of 2700 (the 60×45 pixel size). Results show that the best classification performance is obtained with a minimum of 50 projection directions.

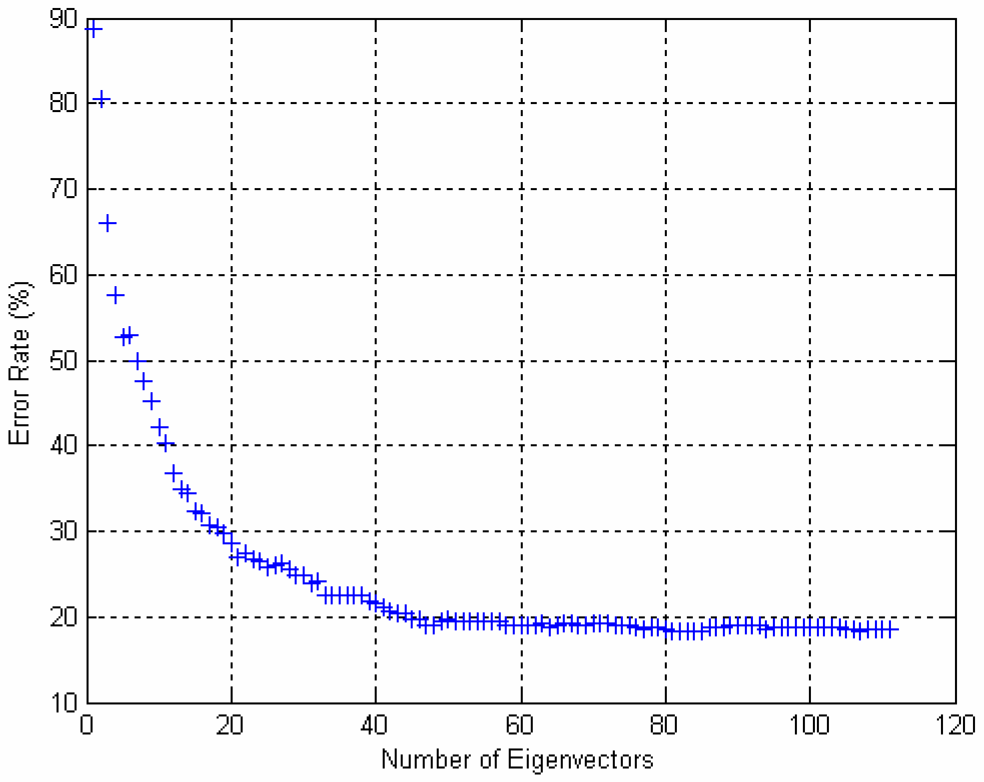

Figure 33. Error Rate vs. the Number of Eigenvectors used in PCA Classification.

C. INCREASING NUMBER OF CLASSES

We began our investigations by increasing the number of classes to observe various PCA-based and LDA classification schemes. As mentioned earlier in Sections A and B of this chapter, the minimum number of eigenvectors is 50 and the fewest number of repetitions for k-fold cross validation is 200, in order to yield satisfactory results. Table 2 lists the mean error rates obtained for various PCA-based schemes and the LDA implementation for 14 to 20 classes in increment of 1 class at a time.

Using the results obtained from Figure 33, we used the top 50 eigenvectors of the PCA-based schemes, which results in mean error rate between 18% and 20% for qualitative analysis. The following notations are used for various PCA-based schemes and the LDA implementation, and all PCA-based schemes prefixed with "PCA":

- PCA50: using the top 50 eigenvectors;

- PCA50W1: using the top 50 eigenvectors after removing the top eigenvector;
- PCA50W2: using the top 50 eigenvectors after removing the top two eigenvectors;
- PCA50W3: using the top 50 eigenvectors after removing the top three eigenvectors;
- PCA50W4: using the top 50 eigenvectors after removing the top four eigenvectors;
- PCA50W5: using the top 50 eigenvectors after removing the top five eigenvectors; and
- LDA: the LDA implementation.

Table 2 lists and Figure 34 plots the mean error rates in percentages, expressed as a function of the number of classes. Results shows that the best performing PCA-based scheme, in the $14-20$ class number range, is PCA50W3 for 14 and 15 classes, PCA50W4 for 16 and 17 classes, PCA50W2 for 18 and 19 classes, and PCA50 for 20 classes, respectively. Results illustrate that removing first few top eigenvectors does not improve the PCA-based algorithm consistently, as we noted in a small database environment in the earlier Pereira study [1]. Results also show that the LDA scheme performance remains consistently better with under 1% error rate for this study.

# of Classes	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
14	17.88	13.19	12.31	8.51	10.03	14.31	0.62
15	16.38	11.30	11.04	8.24	9.08	12.13	0.42
16	15.61	12.29	11.36	11.93	10.41	11.08	0.34
17	14.87	12.45	12.25	16.23	11.64	12.29	0.29
18	14.22	13.13	12.71	17.44	12.94	13.56	0.58
19	13.95	12.77	12.02	16.52	12.78	14.15	0.67
20	13.05	13.34	13.08	18.21	13.79	14.99	0.76

Table 2. Mean Error Rate (%) for 14-20 Classes.

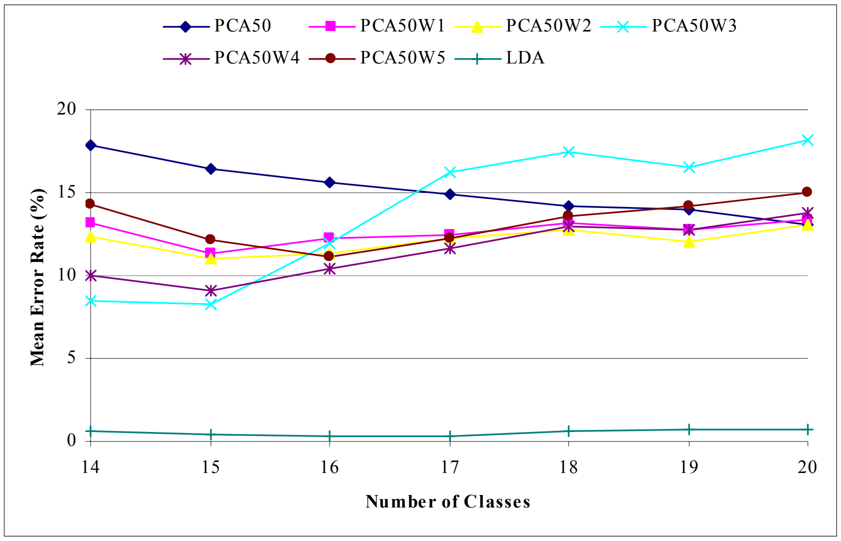

Figure 34. Mean Error Rate (%) Plot for 14-20 Classes.

Table 3 and Figure 35 present the median error rates in percentages, expressed as a function of the number of classes. Note that median error rates were also considered to evaluate the classification performances while minimizing the contribution of significant outliers, which would bias mean error rate results. Results shows that the smallest PCA-based classification error rate is obtained with PCA50W3 for 14 and 15 classes, PCA50W1 for 16 classes, PCA50 for 17 to 20 classes, respectively. The LDA-based classification error rate is above 99.9% for the range of classes considered here. Next, we examined the PCA-based and LDA schemes by increasing the numbers of classes in increments of 5 classes up to all 50 classes, to investigate the impact the number of classes has on overall classification performances.

# of Classes	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
14	12.67	11.13	12.15	5.21	7.29	12.50	0.10
15	12.04	8.38	7.21	3.33	5.88	8.29	0.08
16	9.25	6.83	8.58	10.35	7.00	9.00	0.04
17	4.17	5.54	6.50	11.54	6.71	11.88	0.00
18	4.63	7.54	9.71	13.90	9.17	12.15	0.02
19	5.58	5.75	7.83	15.08	7.13	12.42	0.04
20	5.23	6.71	10.19	14.98	10.69	12.42	0.06

Table 3. Median Error Rate (%) for 14-20 Classes.

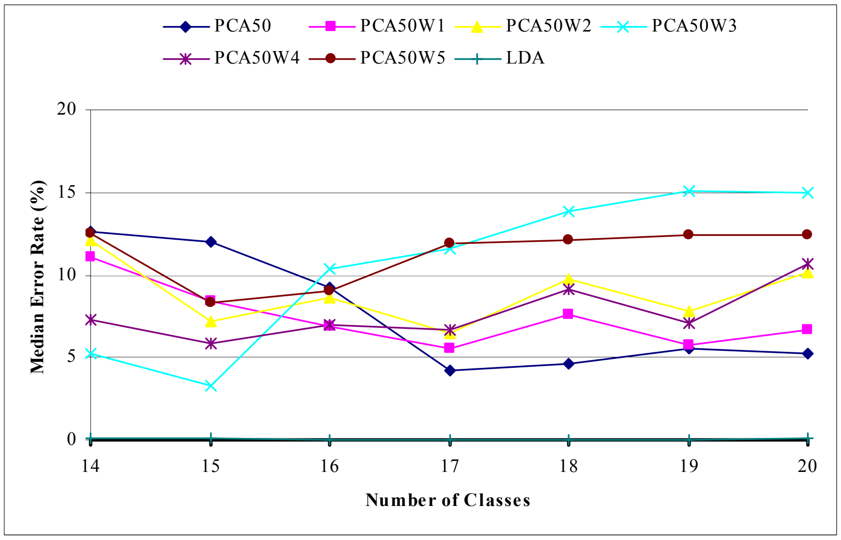

Figure 35. Median Error Rate (%) Plot for 14-20 Classes.

Table 4 lists and Figure 36 plots the mean error rates in percentages, expressed as a function of the number of classes for various PCA-based implementations and the LDA scheme. Results show that PCA50W3 has the smallest error rate for the 15-classes case. Results also show that the best PCA-based performance obtained when the class number exceeds 20 is PCA50. The classification performance again degrades by removing the top eigenvectors. Results also show that the error rates obtained for the LDA and all PCA-based schemes increase as well, as the number of classes increases. This trend is to be expected as an increase in the number of classes increases the number of the images in the dataset, and with it the likelihood of misclassification.

# of Classes	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
15	16.38	11.30	11.04	8.24	9.08	12.13	0.42
20	13.05	13.34	13.08	18.21	13.79	14.99	0.76
25	12.90	16.12	16.56	20.20	16.10	17.26	1.96
30	14.76	19.40	20.68	23.78	19.13	21.59	2.60
35	17.35	22.19	23.85	28.48	22.62	25.21	3.60
40	19.25	23.78	26.01	29.15	24.86	27.58	4.08
45	21.48	25.97	27.31	29.44	27.11	30.96	4.78
50	22.39	26.66	27.51	30.02	28.77	32.57	5.39

Table 4. Mean Error Rate (%) as a Function of the Number of Classes (15-50 Classes in Increment of 5 Classes).

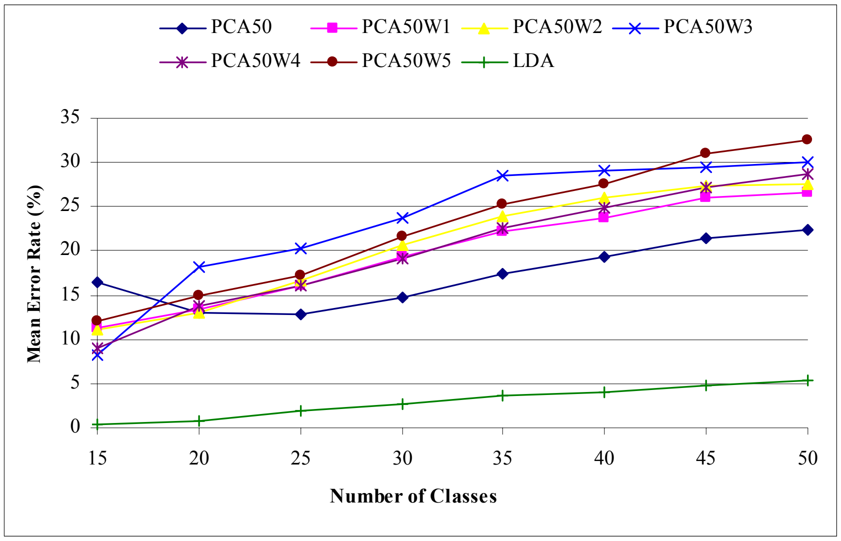

Figure 36. Mean Error Rate (%) as a Function of the Number of Classes (15-50 Classes in Increment of 5 Classes).

Table 5 summarizes and Figure 37 plots median classification error rates, expressed as a function the number of classes considered in the recognition. Results show that the best PCA-based scheme is PCA50W3 for the15-class case. Results also again show that removing the first top few eigenvectors degrade performances results, and that the smallest error rate is obtained consistently with PCA50 when the number of classes is 20 or above. Finally results also show the consistently better classification performance obtained for the LDA implementation.

# of Classes	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
15	12.04	8.3	7.21	3.33	5.88	8.29	0.08
20	5.23	6.71	10.19	14.98	10.69	12.42	0.06
25	6.04	11.88	12	15.88	17.33	14.79	0.33
30	9.73	15.42	17.73	20.9	18.27	18.73	1.4
35	14.29	16.5	22.54	27.54	20.97	25.71	2.04
40	16.06	19.17	25.33	29.52	24.85	27.48	2.71
45	18.13	25.17	27.42	29.5	27.54	29.92	3.79
50	19.25	26.66	27.51	30.02	28.75	32.57	4.04

Table 5. Median Error Rate (%) as a Function of the Number of Classes (15-50 Classes in Increment of 5 Classes).

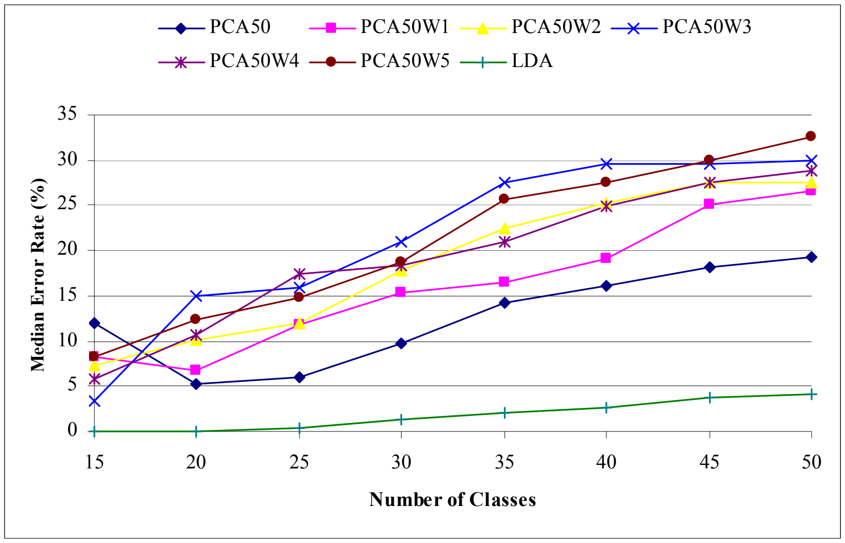

Figure 37. Median Error Rate (%) as a Function of the Number of Classes (15-50 Classes in Increment of 5 Classes).

D. REMOVING CLASS WITH THE HIGHEST ERROR RATE

Simulations showed that Class #13 had a significantly higher error rate than other classes in the database, and specific details regarding this class performance are included in Appendix B. We could find no specific reason for that specific class having significantly lower performance as compared to the other classes available for the study. Thus, we investigated the specific impact this "outlier" class had on overall results by removing it from the database and re-evaluating error rates.

Table 6 lists and Figure 38 plots mean error rates in percentages, expressed as a function of the number of classes, with class #13 removed from consideration. Results in Figure 38 indicate a small improvement in the average error rate by removing Class #13. Results also show classification performance degradations are observed by removing the first top eigenvectors when the class number size reaches 25 or higher. In addition, LDA performances are again significantly better than those obtained with any of the PCA-based.

# of Classes	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
15	11.50	8.74	8.77	12.18	8.78	11.36	0.33
20	11.05	10.17	11.05	15.49	11.72	14.12	0.79
25	9.88	12.61	14.23	18.41	14.70	17.66	1.57

Table 6. Mean Error Rate (%) as a Function of the Number of Classes (15-25 Classes); Outlier Class #13 Removed from the Database.

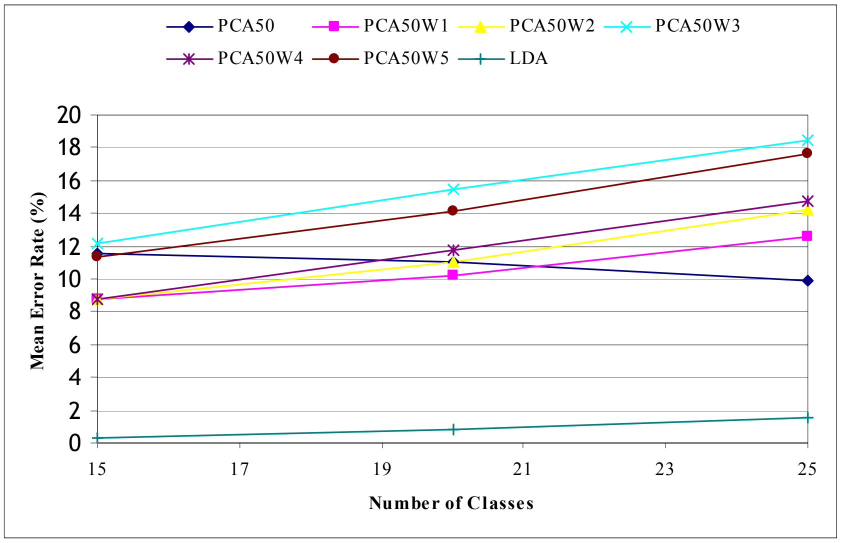

Figure 38. Mean Error Rate (%)as a Function of the Number of Classes (15-25 Classes); Outlier Class #13 Removed from the Database.

Table 7 lists and Figure 39 plots the median error rates for the same experiment. Again PCA50W1 is best the performing PCA-based implementation for 15-class case. Thereafter, only PCA50 is suitable for class size greater than or equal to 20 classes. LDA has less than 0.4 % error rates.

# of Classes	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
15	7.87	6.50	8.10	10.80	6.65	10.51	0.04
20	4.29	7.77	8.56	11.67	7.98	12.79	0.02
25	4.88	8.92	11.42	15.54	12.58	17.17	0.33

Table 7. Median Error Rate (%)as a Function of the Number of Classes (15-25 Classes); Outlier Class #13 Removed from the Database.

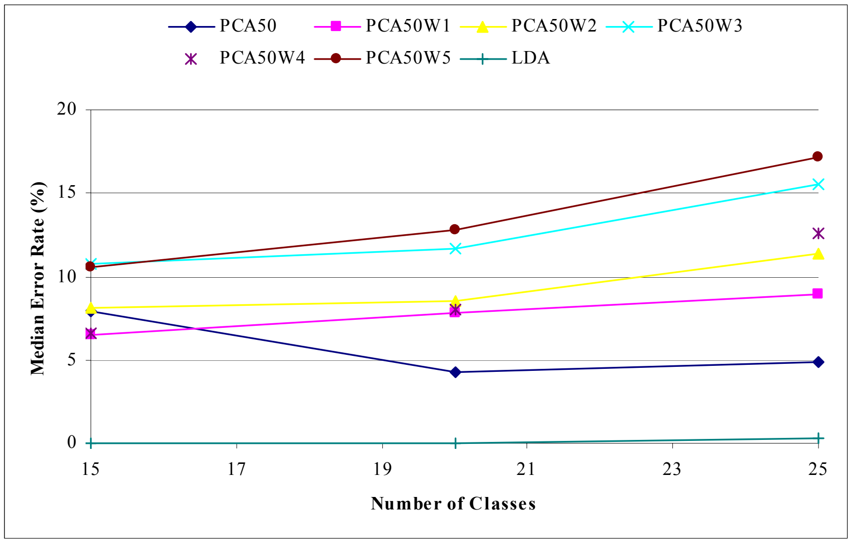

Figure 39. Mean Error Rate (%) as a Function of the Number of Classes (15-25 Classes); Outlier Class #13 Removed from the Database.

E. OVERALL PCA-BASED CLASSIFICATION RESULTS

For reference, Figure 40 plots the average error rates obtained with a direct implementation (no PCA or LDA implementation). For the direct implementation, we computed the class-specific centroids using the original training set without dimensionality reduction. Next, the minimal distance classifier was applied to each testing image and compared to all the class-specific centroids. In practice, we want to avoid the direct implementation since it requires the most computational cost and it does not necessarily produce the most accurate results, and this is why we have to develop a better classification scheme.

Results show that PCA-based error rates appear to stabilize by selecting around the top 50 eigenvectors (Figure 33). Therefore, we investigated the specific PCA-based classification performances obtained by selecting all, the top 80, and top 50 eigenvectors. The 900 iteration cross validations scheme was applied to all PCA-based implementations. The reason we used 900 repetitions was to ensure we trained most of the training images each time we selected different testing/training combinations for the entire k-fold cross validation procedure. Recall that all PCA-based classifications were results from the maximum dimensionality reduction of 900, which corresponds to 900 training images. We examined different PCA-based classification, including removing the first few top eigenvectors for completeness, and applied k-fold cross validation. Results are shown in Figures 41-52:

- PCAA: using all eigenvectors (Figure 41);
- PCAW1: using all the eigenvectors after removing the top eigenvector (Figure 42);
- PCAW2: using all the eigenvectors after removing the top two eigenvectors (Figure 43);
- PCAW3: using all the eigenvectors after removing the top three eigenvectors (Figure 44);
- PCA80: using the top 80 (i.e., the eigenvectors associated with the 80 largest eigenvalues) (Figure 45);
- PCA80W1: using the top 80, after removing the top eigenvector (Figure 46);
- PCA80W2: using the top 80, after removing the top two eigenvector (Figure 47);
- PCA80W3: using the top 80, after removing the top three eigenvector (Figure 48);
- PCA50: using the top 50 (Figure 49);
- PCA50W1: using the top 50, after removing the top eigenvector (Figure 50);
- PCA50W2: using the top 50, after removing the top two eigenvector (Figure 51);
- PCA50W3: using the top 50, after removing the top three eigenvector (Figure 52).

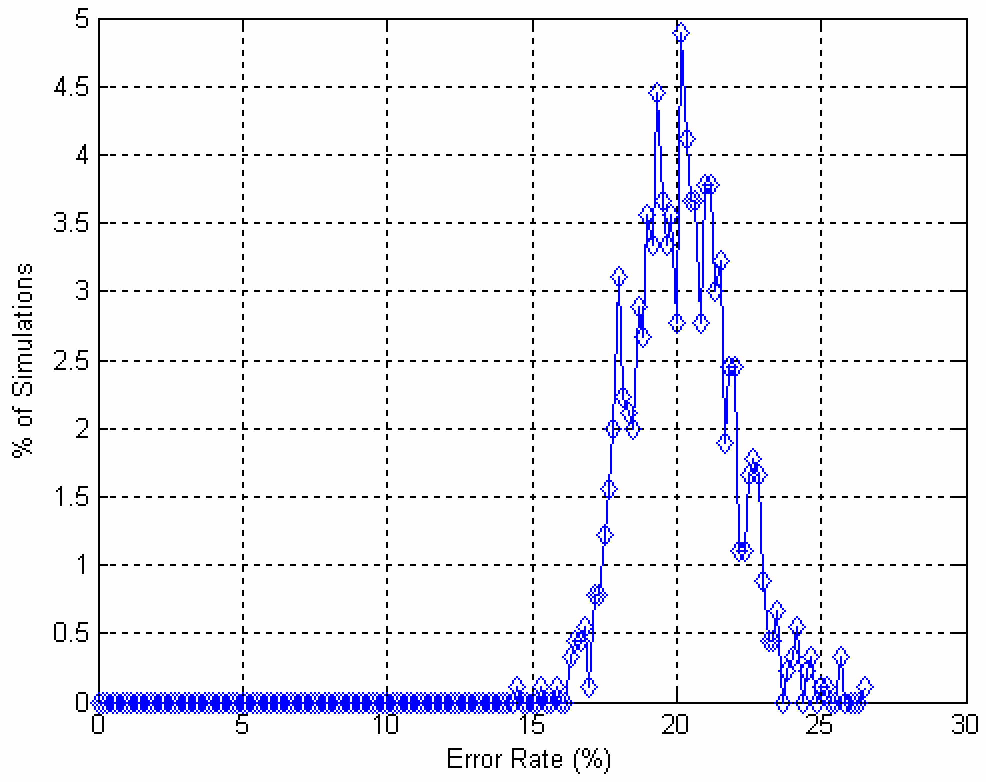

Figure 40. Direct Classification Error Rate (%) Histogram.

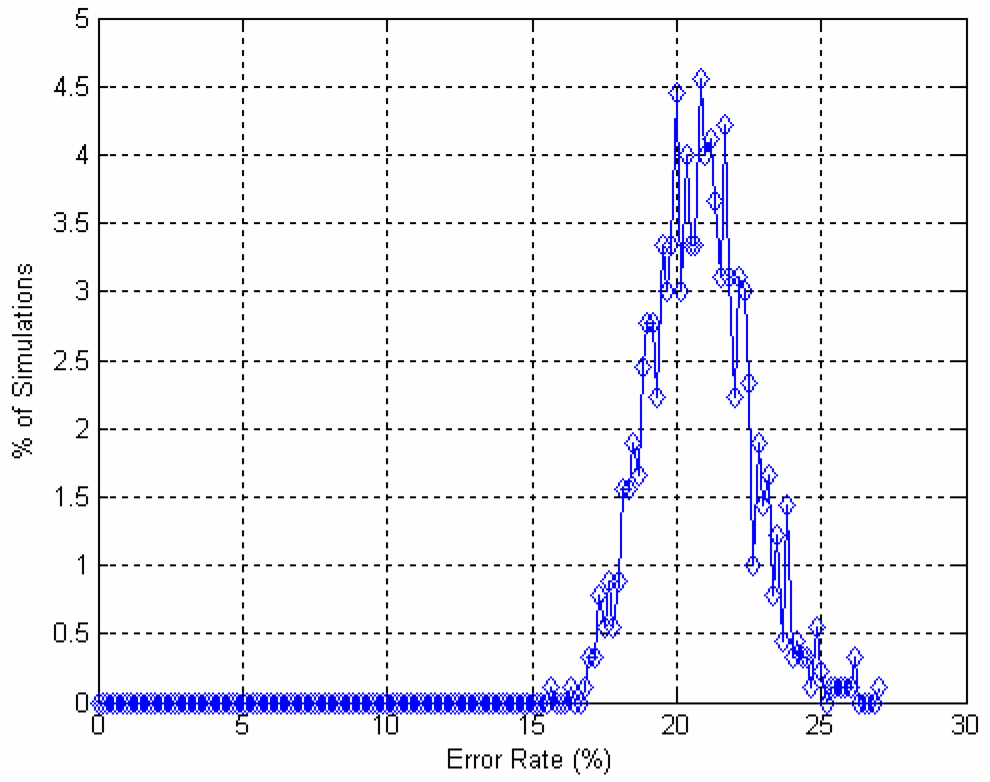

Figure 41. PCAA Classification Error Rate (%) Histogram.

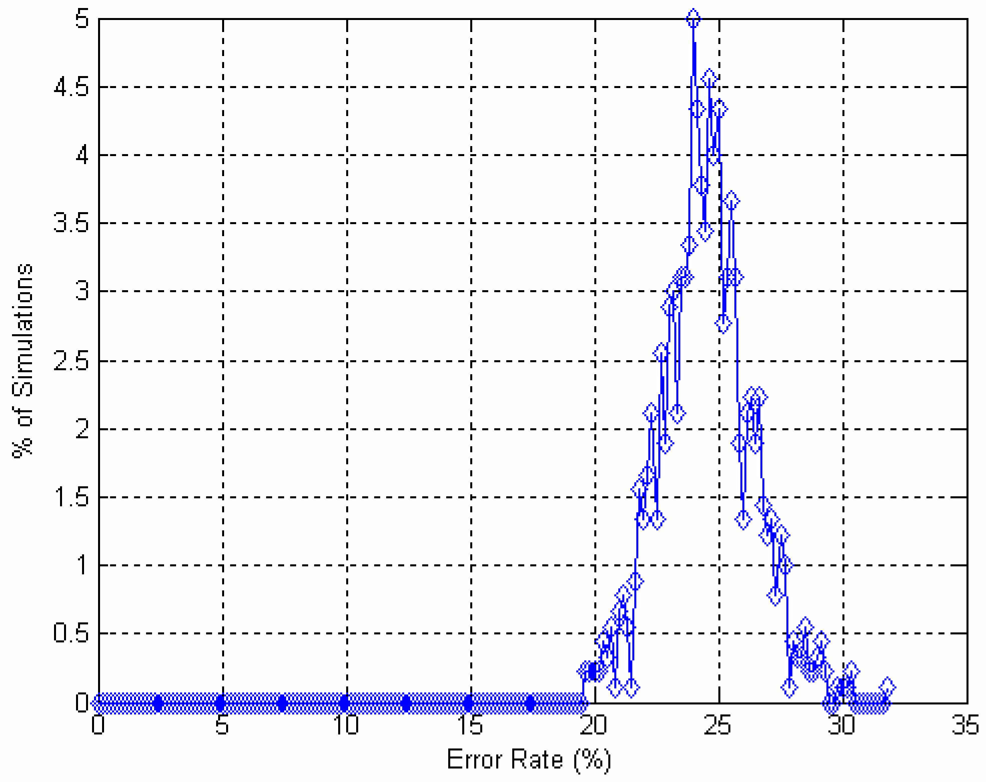

Figure 42. PCAW1 Classification Error Rate (%) Histogram.

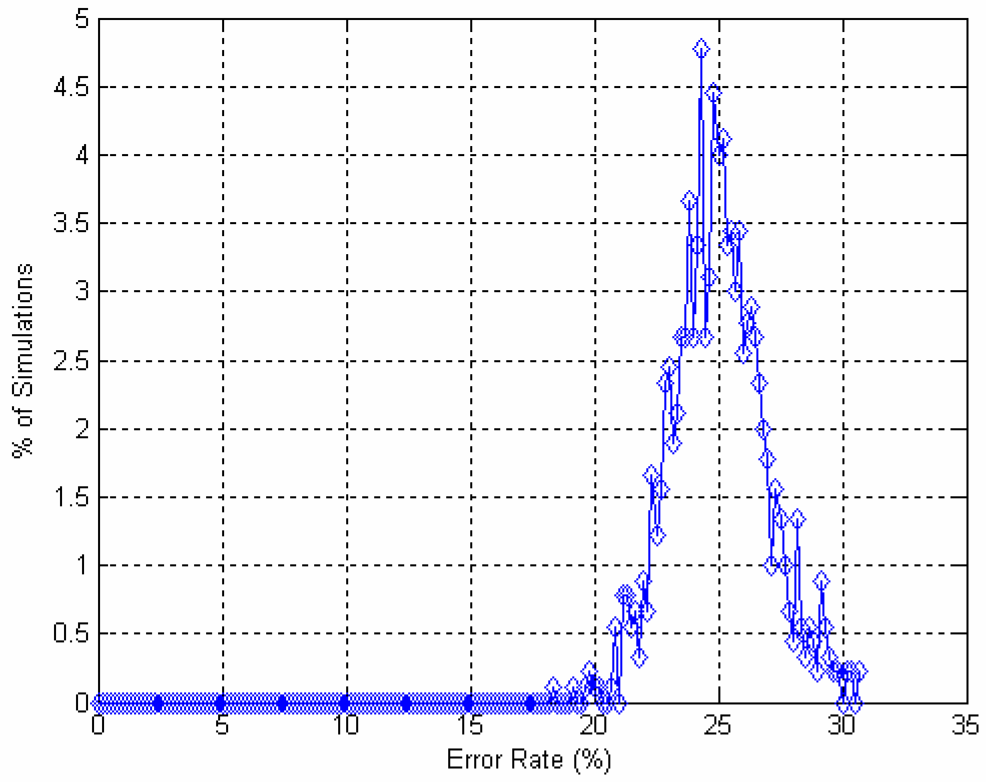

Figure 43. PCAW2 Classification Error Rate (%) Histogram.

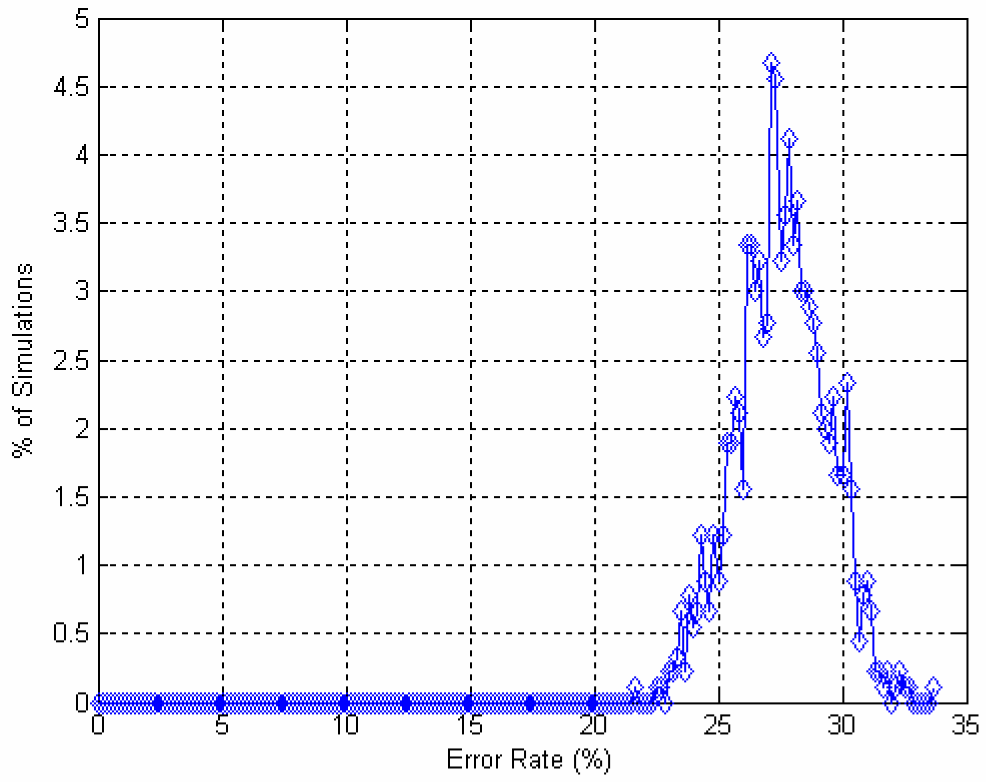

Figure 44. PCAW3 Classification Error Rate (%) Histogram.

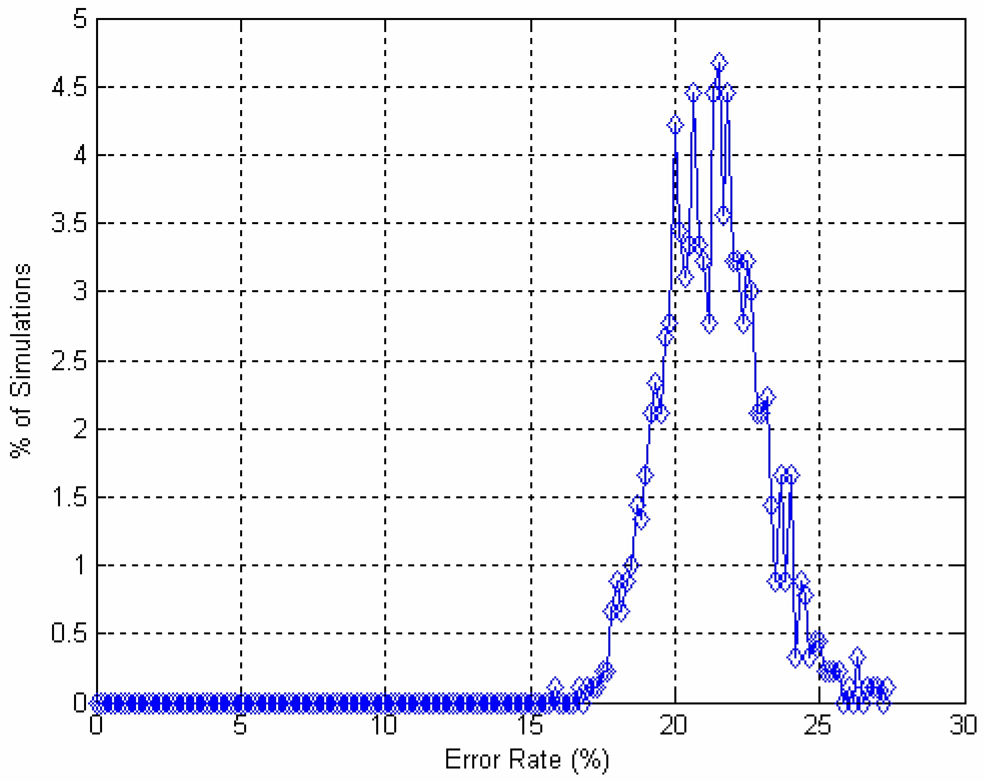

Figure 45. PCA80 Classification Error Rate (%) Histogram.

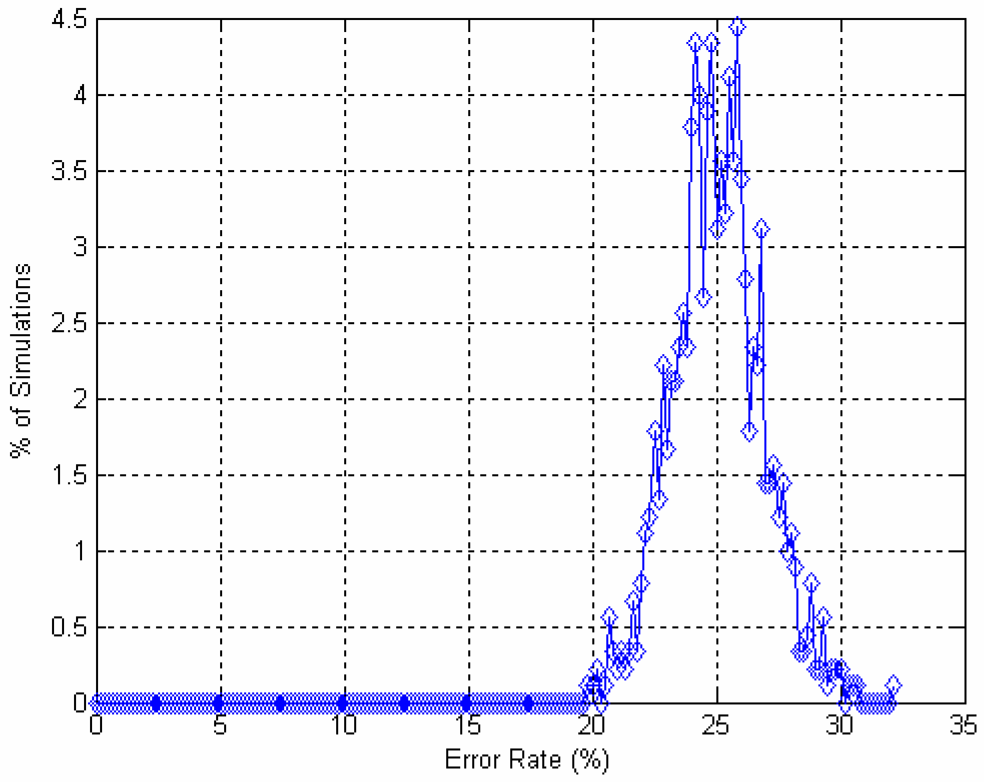

Figure 46. PCA80W1 Classification Error Rate (%) Histogram.

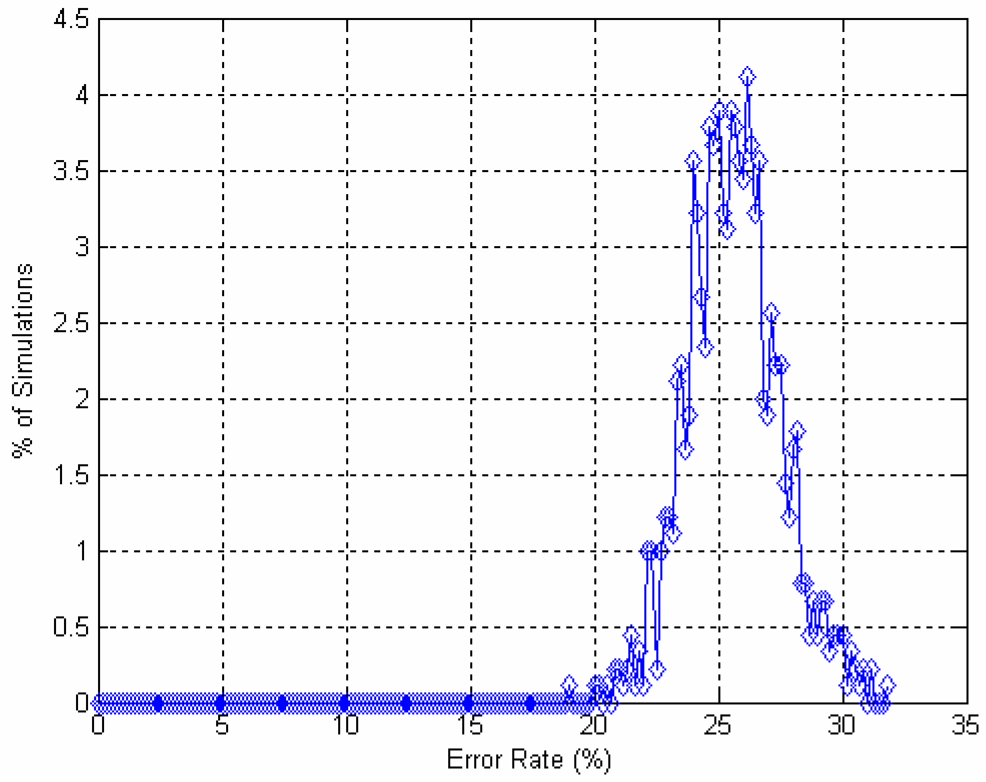

Figure 47. PCA80W2 Classification Error Rate (%) Histogram.

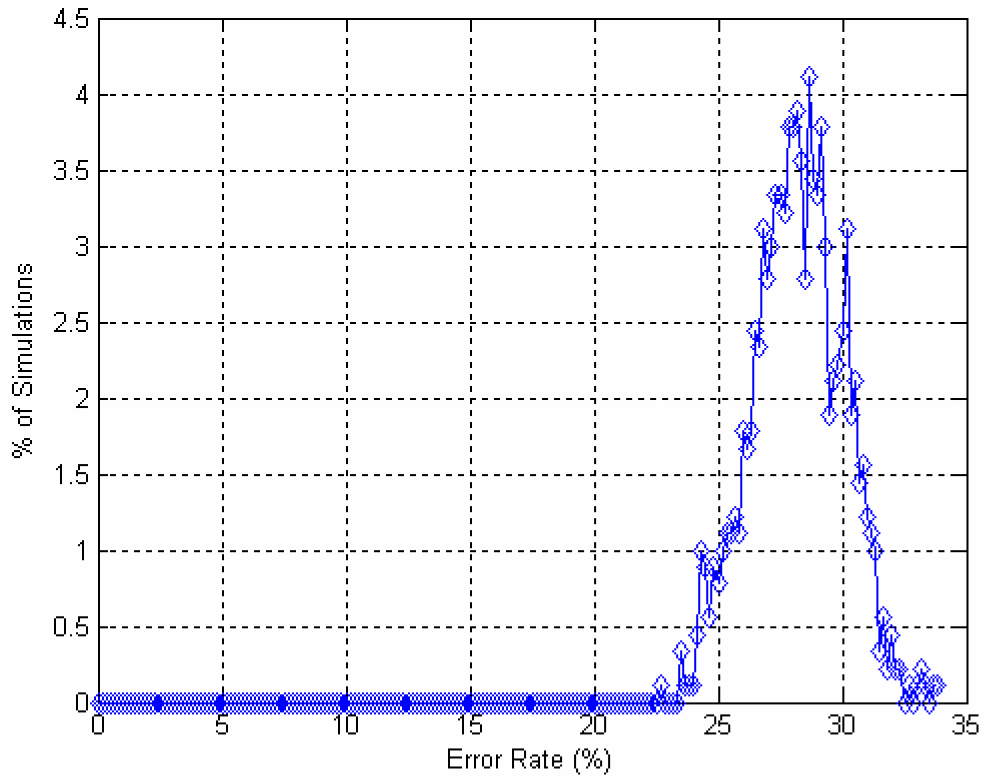

Figure 48. PCA80W3 Classification Error Rate (%) Histogram.

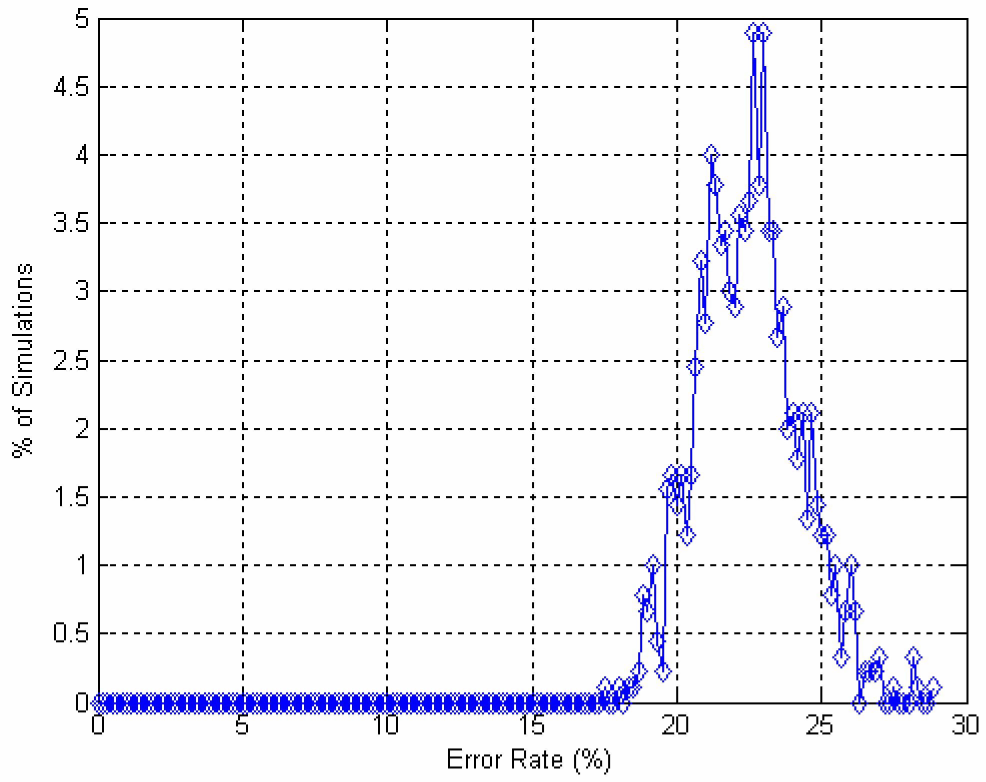

Figure 49. PCA50 Classification Error Rate (%) Histogram.

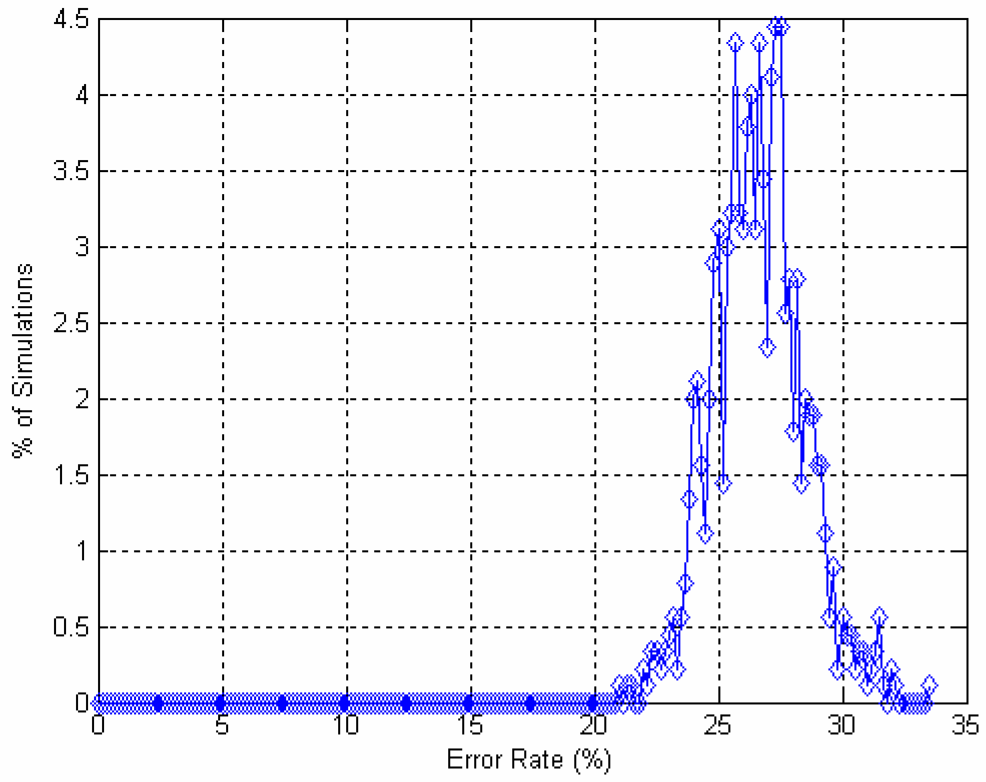

Figure 50. PCA50W1 Classification Error Rate (%) Histogram.

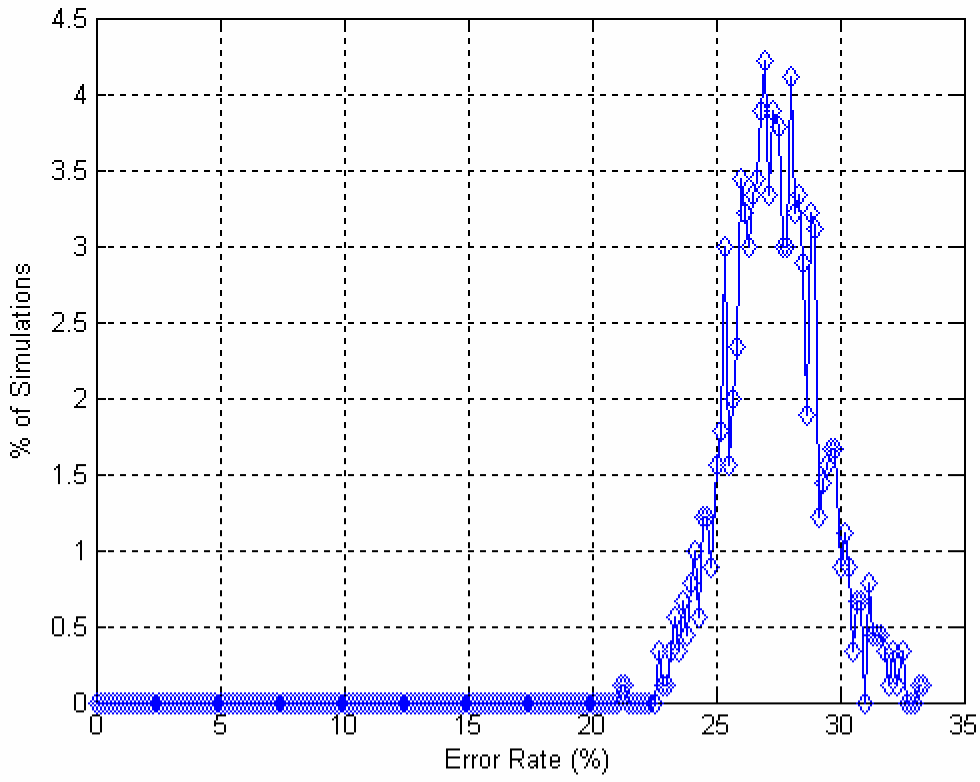

Figure 51. PCA50W2 Classification Error Rate (%) Histogram.

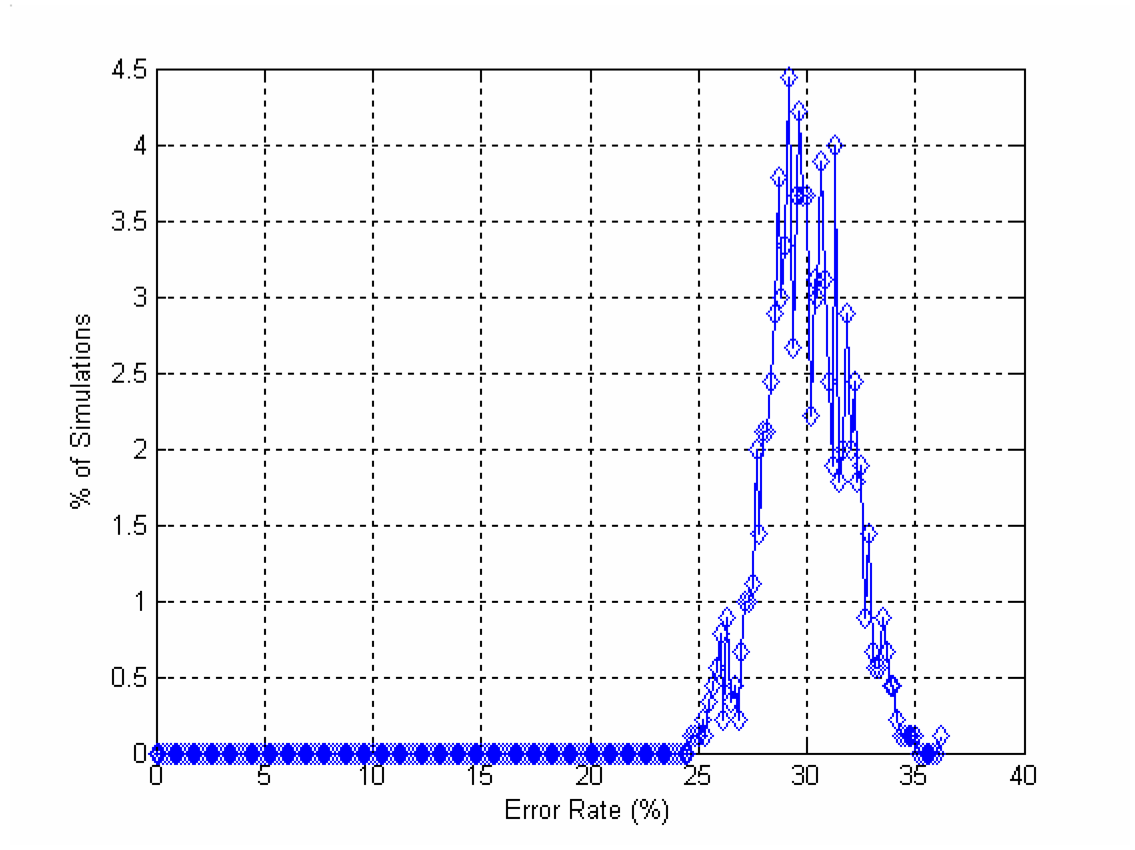

Figure 52. PCA50W3 Classification Error Rate (%) Histogram

Figures 40, 41, 45, and 49 produced similar results as mean error rates are between 20% and 22%. There is no significant degradation on the system performance as we compare the direct classification with performances obtained with PCAA (using all eigenvectors) and PCA80 (using top 80 eigenvectors). We observe a slight performance degradation for the PCA50 implementation.

No significant improvement for removing any of the top three eigenvectors as shown in Figures 42-44. The same behavior occurs in Figures 46-48 and 50-52. As mentioned in Section C and D of this chapter, the PCA-based algorithms must maintain at least top 50 eigenvectors in order to perform classification. Results suggest that the PCA-based algorithms have limitations and no longer produce accurate classifications with larger database. Results show that the mean error rate of the best PCA scheme (PCAA) is 20.94%, followed by PCA80 (21.36%) and PCA50 (22.57%), which is far from what we desire for good classification systems.

F. OVERALL LDA-BASED CLASSIFICATION RESULTS

Direct classification was less than 1% lower in mean error rate than the best PCA-based scheme; however, the direct classification requires much more computational load than PCA-based and LDA schemes. This result illustrates the fact that PCA is best matched for dimensionality reduction, but is not specifically designed for classification application.

Figure 53 shows classification performances obtained with the LDA-based Fish-erface implementation. Results show that the mean error rate is 5.42%, which is 15.52% less than the best PCA-based classification (PCAA). It is also far better than the direct classification (14.74% less in mean error rate). Remember our training set contains 900 images, so 5.42% error rate means about 49 out of 900 images produced erroneous classi-fication, which is obviously outperforming PCA that yields about 189 erroneous classifi-cations.

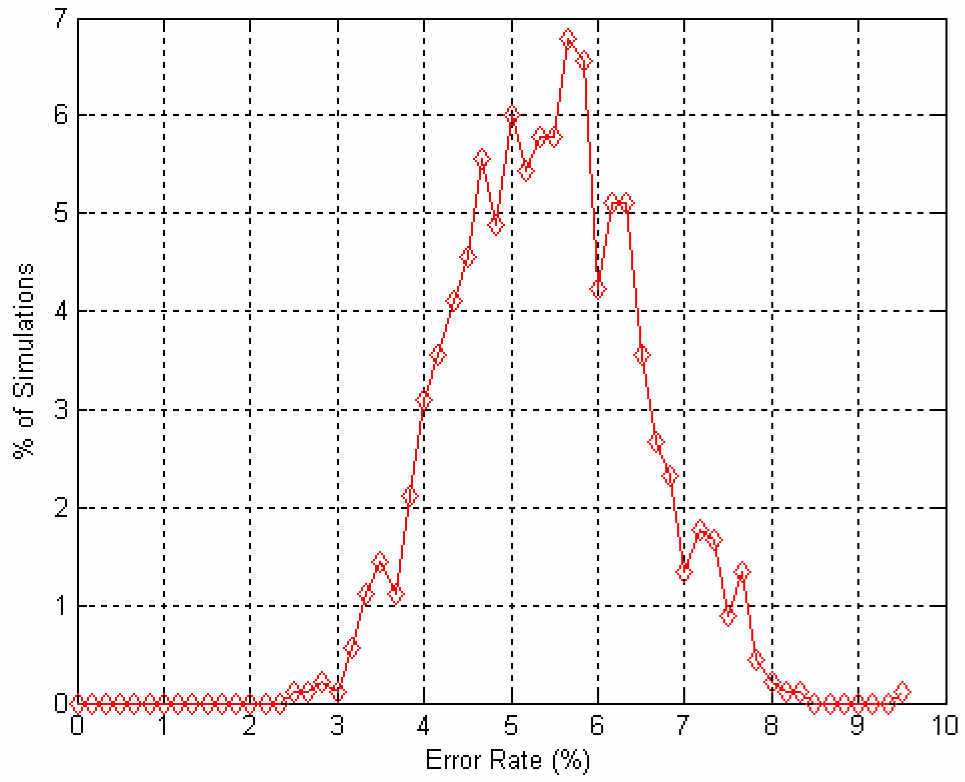

Figure 53. LDA Classification Error Rate (%) Histogram.

Figure 54 plots the distances between a single image of Class #50 and all 50 training set centroids, and can be viewed as a measure of robustness of LDA-based classification implementation. The bar is used to indicate the minimum, mean, and maximum distances from subject #50 to any class-specific centroid. The vertical axis represents the distance between the Class #50 and the class-specific centroids. As shown in Figure 54, the distance separation is smallest in Class #50 that means the images from that class are very close to each other, while being quite well separated from all other classes. Similar outcomes were observed for all other classes.

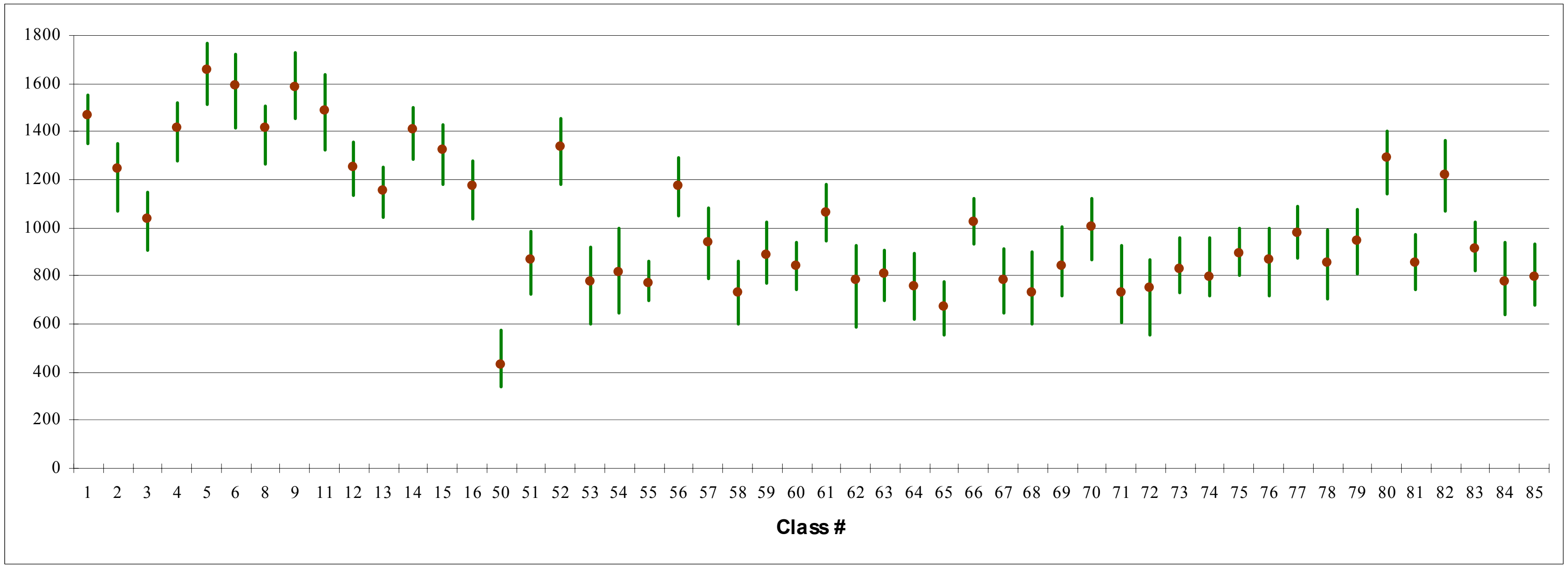

Figure 54. LDA-Based Classification.

Table 8 summarizes the classification performances. The mean error rate obtained for each classification scheme is shown at the bottom of the table for comparison. Results show that All (PCA-based classification using all eigenvectors) has best performance among all PCA-based schemes investigated, followed by PCA80 (using top 80 eigenvectors) and PCA50 (using top 50 eigenvectors). Results also show that degradation in performance when removing the first few top eigenvectors for PCA-based classifications. LDA has the best performance and the lowest mean error rate of 5.42%.

Class #	Direct	PCAA	PCAW1	PCAW2	PCAW3	PCA80	PCA80W1	PCA80W2	PCA80W3	PCA50	PCA50W1	PCA50W2	PCA50W3	LDA
1	45.97	46.34	33.31	24.61	31.33	46.70	33.54	25.51	32.40	47.72	34.23	28.69	34.51	9.52
2	37.76	38.18	36.69	28.00	33.83	38.65	37.14	28.63	34.52	39.73	38.38	30.57	36.43	6.64
3	15.44	16.11	16.08	32.00	26.82	16.78	16.47	33.00	27.88	18.09	17.67	35.40	30.34	2.50
4	14.44	15.52	2.51	0.03	0.05	16.32	2.87	0.06	0.06	18.55	4.19	0.11	0.20	0.00
5	0.03	0.03	15.31	25.45	24.41	0.03	15.38	25.74	24.73	0.03	16.58	26.91	26.55	0.03
6	17.09	17.70	4.75	6.84	7.02	18.23	5.08	7.15	7.40	19.31	6.06	7.33	7.71	0.22
8	11.97	12.21	24.24	35.30	29.40	12.38	24.71	35.65	29.93	12.77	25.72	36.33	31.18	4.18
9	0.43	0.45	0.04	0.00	0.14	0.44	0.04	0.00	0.12	0.50	0.06	0.00	0.18	0.52
11	13.34	14.31	22.44	21.07	21.37	14.73	23.32	21.46	21.69	17.03	26.87	23.28	23.54	0.78
12	3.44	3.57	3.53	2.33	5.61	3.74	3.67	2.69	5.94	4.13	3.91	3.71	6.58	0.19
13	76.91	77.43	85.12	64.33	63.72	78.09	85.95	65.73	64.86	78.94	87.29	68.90	68.56	2.80
14	4.69	4.88	3.83	2.72	3.68	4.96	3.88	2.78	3.71	5.99	4.17	3.17	3.89	0.06
15	0.79	0.94	10.92	11.11	10.75	1.09	11.18	11.07	10.89	2.03	12.86	12.74	11.73	0.01
16	3.88	4.07	2.07	5.38	6.36	4.28	2.17	5.69	6.85	4.73	2.56	6.69	8.57	0.86
50	7.31	7.83	12.03	17.44	23.82	8.23	12.49	17.75	24.54	8.55	14.40	19.02	26.53	4.01
51	2.05	2.17	2.14	4.81	4.83	2.33	2.44	5.03	4.76	3.19	3.13	6.25	6.09	0.50
52	4.24	4.56	9.16	7.38	13.07	4.69	9.38	7.65	13.23	5.31	10.89	9.27	14.40	1.63
53	31.59	31.75	38.64	39.83	37.28	31.84	38.70	40.07	37.07	32.51	38.90	40.54	37.27	14.33
54	1.25	1.39	8.46	10.46	10.78	1.50	8.68	10.60	11.21	1.89	9.75	11.93	12.51	0.31
55	14.11	14.88	42.83	36.90	32.85	15.73	43.82	38.07	33.70	17.43	46.58	40.69	35.60	1.41
56	14.23	14.83	11.58	7.47	17.25	15.63	12.32	8.00	17.87	17.83	14.54	9.67	20.42	4.14
57	14.10	14.71	24.39	21.44	24.23	15.78	25.69	22.67	25.80	18.64	29.90	26.14	29.32	4.38
58	34.92	35.22	38.47	43.02	46.29	35.52	38.77	43.35	46.63	35.77	40.32	43.77	47.08	12.76
59	13.27	14.02	16.41	18.90	21.05	14.22	16.57	19.18	21.29	15.18	17.19	19.46	22.32	6.58
60	39.15	39.75	53.43	50.89	49.36	40.51	53.93	51.57	49.91	42.21	55.80	54.40	51.91	13.33
61	6.53	6.68	10.98	9.39	12.99	6.87	10.80	9.59	13.07	7.06	11.15	9.75	13.85	1.51
62	11.07	10.93	15.69	19.15	21.23	11.39	16.05	19.60	21.78	11.98	17.19	20.78	22.62	10.94
63	16.83	17.20	36.94	41.80	61.52	17.42	37.31	42.07	62.01	18.71	39.29	43.52	64.25	5.99
64	31.35	32.18	41.24	40.76	40.60	32.52	42.24	42.21	41.65	34.68	45.40	45.71	45.50	11.97
65	25.16	26.15	17.06	19.73	25.73	26.64	17.48	20.24	26.37	28.12	18.76	21.28	28.16	4.31
66	16.77	17.64	20.47	34.66	60.92	18.53	21.79	36.43	62.20	20.04	24.02	40.55	64.57	6.83
67	12.94	13.77	19.09	22.55	27.75	14.77	20.37	23.21	28.57	16.86	23.12	25.32	30.94	1.48
68	50.09	51.05	58.94	40.10	41.86	51.97	59.93	40.86	43.07	54.02	62.68	41.97	43.69	17.63
69	29.74	30.78	32.68	39.24	36.71	31.98	33.96	40.50	38.33	34.50	36.95	43.53	42.21	5.57
70	24.57	25.59	31.02	28.70	16.32	26.04	31.66	29.81	16.94	28.39	34.77	34.30	20.67	3.22
71	20.36	20.38	26.43	23.81	29.37	20.37	26.48	24.01	29.46	20.18	26.19	24.05	29.56	2.57
72	21.69	22.25	32.26	35.62	32.01	22.53	32.13	36.44	32.69	23.49	32.94	38.13	34.17	9.42
73	54.49	55.83	48.24	44.37	47.26	56.68	49.82	45.73	49.41	57.80	52.57	50.30	53.83	7.50
74	30.99	32.58	15.19	21.13	24.00	33.45	15.88	22.29	25.24	35.35	18.48	26.40	29.37	2.76
75	6.38	7.10	15.82	23.31	26.75	7.51	16.48	24.27	27.44	8.77	19.36	27.23	30.85	3.49
76	30.35	31.85	44.64	39.64	53.17	33.19	45.94	41.10	55.21	36.70	48.16	43.28	58.35	9.26
77	21.02	22.23	37.71	40.35	54.44	22.56	38.54	41.07	55.26	24.16	40.29	42.76	57.66	8.38
78	17.46	18.38	19.77	18.69	11.38	19.00	20.26	19.56	11.72	20.03	21.66	21.19	12.32	10.04
79	27.94	28.62	36.23	28.57	30.55	29.35	36.79	29.60	31.42	30.36	38.44	31.93	33.56	4.95
80	25.72	26.12	27.85	29.32	31.95	26.30	27.84	29.36	31.94	26.96	28.28	29.73	32.25	3.76
81	36.22	36.67	34.10	35.21	46.28	37.14	34.38	35.76	46.87	37.82	34.99	36.81	48.23	16.41
82	11.04	11.29	14.54	12.67	12.01	11.42	14.72	12.76	12.06	12.07	16.07	13.71	13.65	1.14
83	28.63	29.96	39.61	34.90	40.94	30.94	40.34	35.45	41.73	32.57	41.42	37.01	43.32	17.52
84	13.00	14.14	8.29	12.47	20.45	14.77	9.10	12.97	21.19	15.83	9.91	14.10	22.47	9.78
85	15.16	16.71	23.74	35.72	26.52	17.33	24.53	36.30	27.04	18.97	26.71	39.02	29.28	2.94
mean	**20.16**	**20.78**	**24.54**	**24.99**	**27.56**	**21.26**	**25.06**	**25.61**	**28.19**	**22.47**	**26.61**	**27.35**	**29.97**	**5.42**
median	**16.11**	**16.96**	**21.45**	**24.21**	**26.63**	**17.38**	**22.56**	**24.89**	**27.24**	**18.84**	**24.87**	**27.07**	**29.46**	**4.07**
Class #	**Direct**	**PCAA**	**PCAW1**	**PCAW2**	**PCAW3**	**PCA80**	**PCA80W1**	**PCA80W2**	**PCA80W3**	**PCA50**	**PCA50W1**	**PCA50W2**	**PCA50W3**	**LDA**

Table 8. Error Rate (%) Per Class.

G. UNFOCUSED INFRARED CAMERA LENS EFFECTS

The infrared camera lens was intentionally unfocused for one image of each set to investigate optical malfunction impacts on the uncooled thermal sensor performance. An additional image was obtained from each subject with neutral facial expression facing straight at the camera (Mark #5 in Figure 7). The infrared camera was manually unfocused to reduce the resolution and to blur out the images. Figure 55 and 56 show a sample unfocused image and its corresponding cropped image. The cropped image replaced the original focused one with the same facial expression and head orientation in the database.

The k-fold cross validation was performed, and Table 9 summarizes cross-validation results. Results indicate that there is only less than 1% deviation in mean error rates per class between the original and unfocused datasets. The results also suggest that the infrared cameras can still perform successfully with reduced resolution.

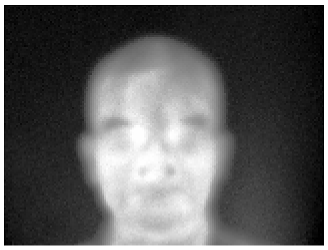
Figure 55. Unfocused IR Image Sample.

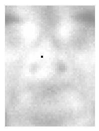
Figure 56. Unfocused IR Cropped Image.

Class #	Direct	PCAA	PCAW1	PCAW2	PCAW3	PCA80	PCA80W1	PCA80W2	PCA80W3	PCA50	PCA50W1	PCA50W2	PCA50W3	LDA
1	45.97	46.38	33.20	24.30	31.17	46.70	33.40	25.14	32.08	47.73	34.09	28.05	34.30	9.45
2	37.76	38.18	37.09	28.60	34.36	38.65	37.50	29.27	34.97	39.64	38.82	31.11	36.82	7.18
3	15.42	16.05	16.09	31.19	25.82	16.64	16.45	32.22	26.97	17.97	17.65	34.69	29.19	2.54
4	14.45	15.54	2.57	0.04	0.04	16.25	2.98	0.06	0.06	18.57	4.33	0.12	0.15	0.00
5	0.03	0.03	14.98	25.35	23.53	0.03	15.13	25.67	23.88	0.03	16.29	26.83	25.80	0.01
6	17.16	17.73	4.71	6.95	6.67	18.28	5.09	7.26	6.89	19.39	6.15	7.50	7.20	0.25
8	11.99	12.19	24.16	35.50	29.50	12.38	24.64	35.75	29.97	12.82	25.70	36.67	31.31	4.32
9	0.41	0.44	0.04	0.00	0.12	0.43	0.04	0.00	0.10	0.48	0.06	0.00	0.15	0.58
11	13.29	14.25	23.44	21.26	21.38	14.69	24.34	21.59	21.69	16.89	27.86	23.32	23.43	0.71
12	3.43	3.56	3.78	2.24	5.33	3.70	3.94	2.55	5.72	4.10	4.14	3.72	6.43	0.20
13	76.82	77.34	85.00	64.39	63.03	78.06	85.72	65.45	64.22	78.87	87.11	68.83	67.98	2.91
14	4.72	4.91	3.82	2.71	3.49	4.96	3.91	2.81	3.56	5.99	4.25	3.13	3.73	0.08
15	0.77	0.96	11.40	11.42	11.05	1.06	11.59	11.51	11.19	2.02	13.42	13.32	12.32	0.02
16	3.91	4.17	2.19	5.39	6.40	4.28	2.24	5.66	6.81	4.69	2.57	6.47	8.53	1.20
50	8.80	9.32	11.94	14.72	22.02	9.68	12.24	15.16	22.58	10.31	13.72	16.05	24.70	6.65
51	4.28	4.36	5.20	7.46	6.97	4.45	5.40	7.65	6.86	5.06	5.90	8.74	7.90	2.18
52	2.70	3.00	6.65	5.41	11.05	3.21	7.10	5.64	11.25	3.57	8.08	6.82	13.04	2.02
53	28.48	28.77	36.87	38.42	34.93	28.74	36.91	38.55	34.96	29.60	37.22	38.88	35.15	13.02
54	1.18	1.36	8.40	9.97	10.71	1.42	8.65	10.14	11.20	1.90	9.86	11.69	12.48	0.31
55	10.83	11.61	44.11	36.93	31.63	12.32	45.33	37.98	32.63	13.80	48.69	40.88	34.98	1.44
56	12.68	13.34	12.55	8.91	20.48	13.74	13.13	9.35	21.05	15.79	15.34	11.02	22.74	3.94
57	15.88	16.44	23.57	19.72	22.62	17.57	24.81	20.93	24.29	19.87	28.51	24.87	28.51	3.31
58	39.10	39.45	39.31	46.46	49.06	39.73	39.79	46.87	49.43	40.23	41.07	47.34	49.93	11.35
59	12.06	12.88	13.92	16.53	20.46	12.90	14.11	16.81	20.69	13.82	14.70	16.97	21.65	6.04
60	39.83	40.53	53.76	52.90	50.27	41.07	54.41	53.80	51.20	42.52	56.32	56.40	52.91	13.92
61	8.74	8.91	13.66	12.32	15.80	8.85	13.57	12.30	15.80	9.17	14.04	12.21	16.77	2.06
62	11.59	11.46	15.30	18.64	20.44	11.94	15.70	19.15	20.91	12.53	16.45	20.19	21.65	10.84
63	17.21	17.61	37.10	42.00	61.25	17.70	37.43	42.32	61.84	18.85	39.40	43.70	64.05	6.56
64	31.09	31.83	41.11	40.81	39.68	32.39	41.88	42.19	40.75	34.47	44.95	45.57	44.30	11.95
65	23.49	24.55	12.13	17.13	21.44	24.98	12.48	17.62	21.84	26.63	13.40	18.68	23.49	4.19
66	17.37	18.21	21.08	35.52	57.59	18.82	22.49	36.87	58.52	20.67	24.71	40.81	60.83	8.10
67	12.81	13.41	20.05	23.42	29.35	14.47	21.14	24.52	30.78	16.54	24.25	27.24	33.44	2.67
68	48.49	49.56	57.57	39.93	41.19	50.21	58.39	40.57	41.94	52.10	61.11	41.60	42.59	17.98
69	28.32	29.38	31.64	38.82	36.28	30.44	32.92	39.96	37.91	33.12	35.79	42.93	41.69	5.82
70	24.19	25.17	31.23	28.55	16.56	25.73	31.79	29.53	17.16	28.18	35.39	34.30	20.74	3.55
71	20.20	20.23	26.59	24.22	29.44	20.24	26.65	24.44	29.48	20.11	26.44	24.58	29.58	2.45
72	22.01	22.86	30.78	34.19	31.44	22.97	30.61	35.14	32.17	23.97	30.94	36.52	32.96	10.60
73	57.16	59.56	47.95	43.21	45.09	60.90	49.68	44.77	46.87	63.48	54.31	50.82	53.09	7.06
74	30.26	31.71	14.19	17.34	19.82	32.46	14.81	18.56	21.00	34.01	17.14	22.13	24.75	2.73
75	6.21	6.87	15.89	23.74	27.20	7.08	16.57	24.89	28.01	8.24	19.45	27.55	31.55	3.13
76	32.04	33.48	42.72	38.21	51.01	34.56	43.72	39.05	53.08	37.20	45.64	40.48	55.64	10.15
77	22.10	23.07	38.27	40.68	54.00	23.20	38.78	41.32	54.69	24.33	40.32	42.74	57.00	8.31
78	17.96	18.80	19.63	18.82	11.42	19.05	20.06	19.54	11.76	20.32	21.41	21.22	12.07	10.35
79	27.44	28.08	36.39	27.93	30.16	28.80	37.18	28.85	31.14	30.07	38.64	31.36	33.37	4.84
80	25.57	25.84	27.92	29.65	32.05	26.08	27.85	29.65	32.03	26.63	28.19	30.07	32.28	4.00
81	38.17	38.57	36.30	35.63	46.53	39.10	36.68	36.35	46.90	39.69	37.30	37.20	48.16	15.87
82	10.54	10.69	16.63	14.97	13.71	10.84	17.03	15.03	13.96	11.82	18.83	15.87	15.31	3.03
83	29.48	31.09	40.31	35.71	42.72	32.05	41.06	36.24	43.23	33.63	42.08	37.76	44.78	17.27
84	12.94	14.40	7.29	9.89	19.06	15.07	8.07	10.33	19.54	16.07	8.57	11.20	21.05	9.81
85	17.34	18.73	25.28	36.97	26.97	19.19	25.97	37.56	27.40	20.78	28.11	40.04	29.57	3.94
mean	**20.29**	**20.94**	**24.51**	**24.90**	**27.25**	**21.36**	**25.03**	**25.49**	**27.86**	**22.57**	**26.57**	**27.20**	**29.64**	**5.62**
median	**17.21**	**17.73**	**23.44**	**24.30**	**26.97**	**18.28**	**24.34**	**25.14**	**27.40**	**19.87**	**25.70**	**27.24**	**29.57**	**4.00**
Class #	**Direct**	**PCAA**	**PCAW1**	**PCAW2**	**PCAW3**	**PCA80**	**PCA80W1**	**PCA80W2**	**PCA80W3**	**PCA50**	**PCA50W1**	**PCA50W2**	**PCA50W3**	**LDA**

Table 9. Unfocused Camera Lens Error Rate in % Per Class.

This chapter presented overall PCA and LDA-based classification result. Our results confirmed that the Linear Discriminant Analysis is far superior to the Principal Component Analysis in classification applications. However, results also showed that removing the top eigenvectors did not benefit the PCA-based algorithms when dealing with a larger database. The investigation on unfocused camera lens effects further confirmed the high performance of infrared imagery under uncontrolled environments or not optimal conditions.

VI. CONCLUSIONS

This study investigated face recognition using an uncooled infrared camera with an expanded database. A database containing 420 facial images obtained from 14 volunteers was available from previous study. An additional 1080 images from 36 volunteers were included in the expanded database, resulting in a total of 1500 images. Each subject was required to perform three different facial expressions with 10 head different orientations. Facial expressions considered were neutral sitting, smiling, and pronouncing the "u" vowel. The distance between the subject and the camera was kept constant while permitting a vertical and horizontal angle freedom of 10°. In addition, 36 images (one from each subject from Class #50 to #85) were collected with an intentionally unfocused camera lens for additional scheme analysis.

We developed an automatic image cropping technique to process large amounts of images. In addition, we also implemented an automated data storage and upload mechanism. This study mainly focused on two linear schemes to examine infrared imaging in face recognition. The first linear scheme used was the Principal Component Analysis (PCA). The second linearity approach was the Fisher Linear Discriminant Analysis incorporated with the PCA for dimension reduction and classification. The minimum distance classifier was selected for its simplicity and accuracy in classification applications. Different PCA-based and LDA schemes were compared using a 60/40 k-fold cross validation scheme with 900 iterations.

Results show that uncooled infrared imaging is a viable candidate for face recognition applications, and that the LDA approach is far superior to the various PCA-based classification algorithms investigated in this study, leading to a 94.58% average classification performance. However, PCA and LDA have limitations. PCA cannot exploit extra compression associated with nonlinear relationships. LDA assumes that class means convey most class information; therefore, the LDA favors unimodal data distributions. In addition, LDA cannot separate nonlinearly separable data sets and classes with the same mean [20]. Extensions to nonlinear classification algorithm are currently under study to investigate whether they lead to better classification performances. Finally, results presents here were derived from images collected under very controlled environmental conditions: fixed distance, indoor environment, and limited facial expressions. Extensions to the study should include relaxing such constraints and investigating resulting impacts on classification performances.

APPENDIX A. MATLAB SOURCE CODES

This appendix contains all MATLAB source codes used for this study:

- *AutoCrop,m* automatically crops IR images and converts the resulting cropped images in *.bmp* format;
- *readpgm8.m* read images obtained from IR camera in *.pgm* format;
- *top.m* automatically crops top and bottom portions of IR images;
- *side.m* automatically crops left and right portions of IR images;
- *load_any_img_to_matrix* loads cropped images into a single matrix;
- *pca_dr.m* produces a PCA dimensionality reduction example;
- *pca_ex.m* produces a PCA classification example;
- *lda_ex.m* produces a PCA classification example;
- *pca_vs_lda.m* produces an example to compare PCA and LDA implementation;
- *pca.m* performs PCA implementation;
- *sortem.m* sorts eigenvalues and corresponding eigenvectors in descending order;
- *fld.m* performs LDA implementation;
- *pca_eig*.m performs PCA implementation and plots error rates as a function of number of eigenvectors;
- *variation.m* performs LDA implementation and plots maximum, mean, and minimum distance from class-specific images to class-specific centroids;
- *kcv.m* performs k-fold cross validation.

AutoCrop

```
%-----------------------------------------------------------------------------------------------------
%
% Filename:            AutoCrop.m
% Thesis Advisor:      Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:  Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:              Colin K Lee, Naval Postgraduate School, Monterey, CA, 2004
% Descriptions:        Routine program used to read the binary PGM image data
%                      obtained from IR-160 camera, to crop image automatically to
%                      60x45 pixels, and to convert and rename images file in bmp
%                      format.
% Inputs:
%                      Person:          Class number assigned to each subject
% Outputs:             Cropped image data saved in bmp format
% Function(s) called: readpgm8.m, top.m, and side.m.
%
%-----------------------------------------------------------------------------------------------------
clc
clear
Person=[83] ; % Person numbers contained on the files
N_pictures=10; % number of pictures for each person number
Section=1;
for i=1:length(Person)
  for j=1:N_pictures
     im_num1= num2str(Person(i)); im_num2= num2str(j);im_num3=num2str(Section);
     img_name = strcat(im_num1,'-',im_num2,'-',im_num3,'.pgm');
     A=readpgm8(img_name);    % read pgm files
     B = top(A);
     C = side(B);
     namefinal=strcat(im_num1,'-',im_num2,'-',im_num3,'-a','.bmp');
     imwrite(C,namefinal,'bmp');
  end
end

Section=2;
for i=1:length(Person)
  for j=1:N_pictures
     im_num1= num2str(Person(i)); im_num2= num2str(j);im_num3=num2str(Section);
     img_name = strcat(im_num1,'-',im_num2,'-',im_num3,'.pgm');
     A=readpgm8(img_name);    % read pgm files
     B = top(A);
     C = side(B);
     namefinal=strcat(im_num1,'-',im_num2,'-',im_num3,'-a','.bmp');
     imwrite(C,namefinal,'bmp');
  end
```

```
end

Section=5;
for i=1:length(Person)
  for j=1:N_pictures
      im_num1= num2str(Person(i)); im_num2= num2str(j);im_num3=num2str(Section);
      img_name = strcat(im_num1,'-',im_num2,'-',im_num3,'.pgm');
      A=readpgm8(img_name);    % read pgm files
      B = top(A);
      C = side(B);
      namefinal=strcat(im_num1,'-',im_num2,'-',im_num3,'-a','.bmp');
      imwrite(C,namefinal,'bmp');
  end
end

Section=6;
for i=1:length(Person)
  for j=1:N_pictures
      im_num1= num2str(Person(i)); im_num2= num2str(j);im_num3=num2str(Section);
      img_name = strcat(im_num1,'-',im_num2,'-',im_num3,'.pgm');
      A=readpgm8(img_name);    % read pgm files
      B = top(A);
      C = side(B);
      namefinal=strcat(im_num1,'-',im_num2,'-',im_num3,'-a','.bmp');
      imwrite(C,namefinal,'bmp');
  end
end
```

readpgm8

```
function image = readpgm8(filename)
%-----------------------------------------------------------------------------------------------------
%
% Filename:            readpgm8.m
% Thesis Advisor:      Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:    Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:              Matthew Dailey, 1997
% Modified by:         Colin K Lee, Naval Postgraduate School, Monterey, CA, 2004
% Descriptions:        reads the binary PGM image data obtained from IR-160 camera
% Inputs:
%                      filename:      binary PGM image data
% Outputs:
%                      image:         2-dimensional array of integers
% Function(s) called:  none
%
%-----------------------------------------------------------------------------------------------------
function image = readpgm8(filename)
% Open the file
fid = fopen(filename,'r');
% Parse and check the header information.  No # comments allowed.
A = fgets(fid);
if strcmp(A(1:2),'P5') ~= 1
        error('File is not a raw PGM');
end;
A = fgets(fid);
if strcmp(A(1:2),'#') ~= 0
        error('File is not a raw PGM');
end;
A = fgets(fid);
sizes = sscanf(A,'%d');
w = sizes(1);
h = sizes(2);
A = fgets(fid);
max = sscanf(A,'%d');
tlength = w*h;
if max ~= 255
        error('Cannot handle anything but 8-bit graymaps');
end;
% Read the raw data
[v,count] = fread(fid,inf,'uint8=>uint8'); % this makes v a uint8 instead of double
%if count ~= tlength
% error('File size does not agree with specified dimensions.');
%end;
v=v(1:tlength,1);
```

```
% Pack the column vector v into the image matrix
image = reshape(v,w,h)';
fclose(fid);
```

top

```
%function [CC] = top(B)
%------------------------------------------------------------------------------------------------------
%
% Filename:              top.m
% Thesis Advisor:        Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:      Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:                Colin K Lee, Naval Postgraduate School, Monterey, CA, 2004
% Descriptions:          crops top and bottom portions of image data, results in image
%                        containing facial characteristics above chin and below eyebrows
% Inputs:
%                        B:      image data martix
% Outputs:
%                        CC:     cropped image data matrix .
% Function(s) called:    none
%
%------------------------------------------------------------------------------------------------------

function [CC] = top(B)
rotate_B = rot90(B);
[i,j] = find(rotate_B);
z = [i j];
s = z(1,2);
[rowNum colNum]=size(rotate_B);
for j=s:colNum;
   for i=1:rowNum;
      C(i,j-s+1)=rotate_B(i,j);
   end
end
midP=round((colNum-s)/2);
CC=C(:,midP-18:midP+41);
CC=rot90(CC,3);
return
```

side

```
%function [DD] = side(B)
%-----------------------------------------------------------------------------------------------------
%
% Filename:             top.m
% Thesis Advisor:       Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:     Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:               Colin K Lee, Naval Postgraduate School, Monterey, CA, 2004
% Descriptions:         crops side portions of image data
% Inputs:
%                       B:      image data martix
% Outputs:
%                       DD:     cropped image data matrix in 60x45 pixels
% Function(s) called:   none
%
%-----------------------------------------------------------------------------------------------------

function [DD] = side(B)

[i,j] = find(B);
z = [i j];
s = z(1,2);
[rowNum colNum]=size(B);
for j=s:colNum;
   for i=1:rowNum;
      C(i,j-s+1)=B(i,j);
   end
end

CC = fliplr(C);
[m,n] = find(CC);
zf = [m n];
sf = zf(1,2);
[rowNumf colNumf]=size(CC);
for n=sf:colNumf;
  for m=1:rowNumf;
     DD(m,n-sf+1)=CC(m,n);
  end
end
D = fliplr(DD);
[h k] = size(D);
midP=round(k/2);
DD=D(:,midP-26:midP+18);
Return
```

load_any_img_to_matrix

```
% function [A,T] = load_any_img_to_matrix(C,O,S)
%--------------------------------------------------------------------------------------------------
%
% Filename:            load_any_img_to_matrix.m
% Thesis Advisor:      Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:    Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:              Colin K Lee, Naval Postgraduate School, Monterey, CA, 2004
% Descriptions:        load any image into a single matrix
% Inputs:
%                      C:      Class #
%                      O:      number of images
%                      S:      Section # of specific facial expression
% Outputs:
%                      A:       a single data matrix
%                      T:      Class #
% Function(s) called:  none
%
%--------------------------------------------------------------------------------------------------
function [A,T] = load_any_img_to_matrix(C,O,S)
Person = C;
N_objects = length(O);
N_class=length(Person);
Section = S;
A = [];
T = [];
for i=1:N_class
  for j=1:N_objects
     for k=1:length(Section)
     im_num1= num2str(Person(i));
     im_num2= num2str(j);
     s=num2str(Section(k));
     img_name = strcat(im_num1,'-',im_num2,'-',s,'-','a');
      img= imread(img_name,'bmp');
     [dim1,dim2] = size(img);
     x=reshape(img,dim1*dim2,1);
     A=[A x];  % A contains one image in each column
     T=[T Person(i)];% T contains the class of each image
     end
  end
end
```

pca_dr

```
%-----------------------------------------------------------------------------------------------------
%
% Filename:            pca_dr.m
% Thesis Advisor:      Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:    Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:              M. Cairns
% Modified by:         Colin K Lee, Naval Postgraduate School, Monterey, CA, 2004
% Descriptions:        Routine program to demonstrate the use of PCA in dimensionality
%                      reduction.
% Inputs:              none
% Outputs:             none
% Function(s) called:  none
%
%-----------------------------------------------------------------------------------------------------
clc
clear all;
close all;
% Generate elliptical cloud of data-poitns.
  x(1,:) = randn(1,100);
  x(2,:) = randn(1,100)*3;
% Rotate the cloud for demonstration.
  [p(1,:),p(2,:)] = cart2pol(x(1,:),x(2,:));
  p(1,:) = p(1,:)-pi/3;
  [x(1,:),x(2,:)] = pol2cart(p(1,:),p(2,:));
% Plot data.
figure(1)
  scatter(x(1,:),x(2,:));
  axis equal;
  % Calculate PC's.
  [pc, latent, explained] = pcacov(cov(x'));
% Draw PC's on top of data.
figure(2)
scatter(x(1,:),x(2,:));
  axis equal;
  hold on
  plot([-4 4]*pc(1,1),[-4 4]*pc(2,1),'r-');
  hold on
  plot([-2 2]*pc(1,2),[-2 2]*pc(2,2),'g-');
  hold off;
% Rotate the data to the PC's
  y = (x'*pc)';
% Plot data.
  figure(3)
  scatter(y(1,:),y(2,:));
  axis equal;

  % Calculate PC's, to demonstrate they now lie on the axes.
[pc2, latent, explained] = pcacov(cov(y'));
% Draw PC's on top of data.
figure(4)
scatter(y(1,:),y(2,:));
  axis equal;
  hold on;
  plot([-4 4]*pc2(1,1),[-4 4]*pc2(2,1),'r-');
  hold on
  plot([-2 2]*pc2(1,2),[-2 2]*pc2(2,2),'g-');
  hold off
  %Set the second component of y to zero, reducing the dimensionality to one.
  y(2,:) = 0;
% Transform back to the original data.
  x = (y'*inv(pc))';
% Plot data.
  figure (5);
  scatter(x(1,:),x(2,:));
  axis equal;
```

pca_ex

```
%-----------------------------------------------------------------------------------------------------
%
% Filename:           pca_ex.m
% Thesis Advisor:     Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:   Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:             Diogo Pereira, Naval Postgraduate School, Monterey, CA, 2002
% Modified by:        Colin K Lee, Naval Postgraduate school, Monterey, CA, 2004
% Descriptions:       Routine program to compare PCA in classification
% Inputs:             none
% Outputs:            none
% Function(s) called: none
%
%-----------------------------------------------------------------------------------------------------
close all
clear all
clc
rand('seed',0);
a1=rand(1,50);
a2=rand(1,50);
b1=rand(1,50);
b2=rand(1,50);
c1=[a1;0.5*a2];
c2=[1.5+b1;0.5*b2];
figure(1)
plot(c1(1,:),c1(2,:),'b*')
hold on
plot(c2(1,:),c2(2,:),'ro')
axis([0 2 -1 1])
A=[c1 c2];
[W,m,Amean,EVA]=pca(A,1);
W;
x=linspace(-2,2);
y1=x*W(1,:)+Amean(1);
y2=x*W(2,:)+Amean(2);
plot(y1,y2,'g-')
grid
%example pca not ok
c1=[4*a1;0.5+0.2*a2];
c2=[4*b1;0.2*b2];
```

lda_ex

```
%-----------------------------------------------------------------------------------------------
%
% Filename:            lda_ex.m
% Thesis Advisor:      Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:    Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:              MathWorks
% Modified by:         Colin K Lee, Naval Postgraduate School, Monterey, CA, 2004
% Descriptions:        Routine program to demonstrate the use of LDA in classification
% Inputs:              none
% Outputs:             none
% Function(s) called:  none
%
%-----------------------------------------------------------------------------------------------
clc
clear
close all
c1=[randn(100,1) randn(100,1)+1 randn(100,1)+6];
c2=[randn(100,1) randn(100,1) randn(100,1)];
figure(1)
  scatter3(c1(:,1),c1(:,2),c1(:,3),'filled')
  hold on;
  scatter3(c2(:,1),c2(:,2),c2(:,3),'r','filled')

figure(2)
  scatter(c1(:,1),c1(:,2),'filled');
  hold on
  scatter(c2(:,1),c2(:,2),'r','filled');
  axis equal;
data=[c1;c2];
label=[ones(100,1);zeros(100,1)];
mode=1;
if(any(label==0)),
    label=label+1;
end
cat=length(unique(label));
[n,f]=size(data);
Sw=zeros(f);
Sb=zeros(f);
m=mean(data);
for i=1:cat,

   [r,c]=find(label==i);
   mg=mean(data(r,:));
   ng=length(r);
   Sw=Sw + cov(data(r,:)).*(ng-1);
   Sb=Sb + ng*(mg-m)'*(mg-m) ;
end
[v2,d2]=eig(inv(Sw)*Sb);
drec=2;
A=v2(:,1:drec);
dataLDA=data*A;
data_final=reshape(dataLDA,100,4);
figure(3)
  scatter(data_final(:,1),data_final(:,2),'filled');
  hold on;
  scatter(data_final(:,3),data_final(:,4),'r','filled');
  hold on;
  plot([-10 4]*v2(1,1),[-10 4]*v2(2,1),'g-');
  hold on;
 axis equal;
drec=3;
A1=v2(:,1:drec);
dataLDA1=data*A1;
data_final1=reshape(dataLDA1,100,6);
% Rotate the data to the v's
 y=(data*v2')';
% Plot data.
```

pca_vs_lda

```
%-----------------------------------------------------------------------------------------------------
%
% Filename:           pca_vs_lda.m
% Thesis Advisor:     Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:   Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:             Diogo Pereira, Naval Postgraduate School, Monterey, CA, 2002
% Modified by:        Colin K Lee, Naval Postgraduate school, Monterey, CA, 2004
% Descriptions:       Routine program to compare PCA and LDA in classification
% Inputs:             none
% Outputs:            none
% Function(s) called: pca.m, fld.m
%
%-----------------------------------------------------------------------------------------------------
close all
clear all
clc
% PCA not work
c1=[4*a1;0.5+0.2*a2];
c2=[4*b1;0.2*b2];
figure(2)
plot(c1(1,:),c1(2,:),'b*')
hold on
plot(c2(1,:),c2(2,:),'ro')
 axis([0 4 -.2 .8])
A=[c1 c2];
[W,m,Amean,EVA]=pca(A,1);
W;
x=linspace(-2,2);
y1=x*W(1,:)+Amean(1);
y2=x*W(2,:)+Amean(2);
plot(y1,y2,'g-')
grid
% LDA works
c1=[4*a1;0.5+0.2*a2];
c2=[4*b1;0.2*b2];
figure(3)
plot(c1(1,:),c1(2,:),'b*')
hold on
plot(c2(1,:),c2(2,:),'ro')
axis([0 4 -.2 .8])
A=[c1 c2];
C=[ones(1,50) ones(1,50)*2]
[W,D]=fld(A,C);
W;
```

```
x=linspace(-2,2);
y1=x*W(1,:)+Amean(1);
y2=x*W(2,:)+Amean(2);

plot(y1,y2,'g-')
grid
```

pca

```
%function [W,m,Amean,Ad,EVA]=pca(A,n)
%-----------------------------------------------------------------------------------------------
%
% Filename:           pca.m
% Thesis Advisor:     Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:   Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:             Diogo Pereira, Naval Postgraduate School, Monterey, CA, 2002
% Modified by:        Colin K Lee, Naval Postgraduate school, Monterey, CA, 2004
% Descriptions:       Function computes the pca(Principal Component Analysis) of the
%                     data contained in the array A.
% Inputs:
%                     A:      k x n data matrix contains each image in each column
%                     n:      number of data samples
% Outputs:
%                     W:      contain n eigenvectors, one in each column
%                     m:      fraction of the variance in n eigenvectors
%                     Amean:          mean of the data contained in A in a column
%                     Ad:     data less the mean Amean, all the data in columns
%                     EVA:    contain the eigenvalues corresponding to the eigenvectors
%                             of W on the diagonal
% Function(s) called: sortem.m
%
%-----------------------------------------------------------------------------------------------
function [W,m,Amean,Ad,EVA]=pca(A,n)
clc
clear all
close all

[Adimk,Adimn]=size(A);
if (nargin==1)
    n=Adimn;
 end
if (n>Adimn)
  n=Adimn;
end
% Amean is the mean of A using the collomns as elements
Amean=mean(A,2);
% Ad is the difference between A and Amean
Ad=A - Amean*ones(1,Adimn);
%eigenvectors (columns of Vectors) and eigenvalues (diag of Values)
[EVE,EVA] = eig(Ad'*Ad);
% obtain index of eigenvalues greater than (0.001 times greater eigenvalue)
[I]=find(EVA>(0.001*max(max(EVA))));
EVAC=zeros(size(EVA));
```

```
EVAC(I)=EVA(I);  %EVAC (Eigenvalue Conditioned) will have only values greater
than a minimum value otherwise the value is set to zero

% Obtain n eigenvectors
  %EVAINV is the matrix containing the inverse of the eigenvalues on the diagonal, it
contains zero if the eigenvalue was zero
  % and the inveseof the eigenvalue if the element was larger than zero
  EVAINV=zeros(Adimn); % EVAIN has size n x n since n is the maximum number of
eigenvalues
  EVAINV(I)=1./(sqrt(EVA(I)));
  U=A*EVE*EVAINV;
%Sort the vectors/values according with the absolute value of the eigenvalue EVAC and
eliminates the collumns corresponding
% to zero eigenvalues
[W,EVA]=sortem(U, EVAC);
if (n>size(W,2))
   n=size(W,2);
end

W=W(:,1:n);
temp=diag(EVA);
m=sum(temp(1:n))/sum(temp);
return
```

sortem

```
% function [NV,ND] = sortem(V,D)
%-------------------------------------------------------------------------------------------------------
%
% Filename:             sortem.m
% Thesis Advisor:       Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:     Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:               Diogo Pereira, Naval Postgraduate School, Monterey, CA, 2002
% Modified by:          Colin K Lee, Naval Postgraduate school, Monterey, CA, 2004
% Descriptions:         Function to set threshold value and sort the eigenvalues and
%                       eigenvectors in descending order.
% Inputs:
%                       V:      eigenvectors
%                       D:      eigenvalues
% Outputs:
%                       NV:     sorted eigenvectors
%                       ND:     sorted eigenvalues with values above threshold
% Function(s) called:   none
%
%-------------------------------------------------------------------------------------------------------
function [NV,ND] = sortem(V,D)

% Sorts the columns of V along with the absolute value of the elements of D and
% elimnates the column of V corresponding to zero eigenvalue
dvec = diag(D);    %obtain the values of the diagonal and insert in a vector
%eliminating the column on V corresponding to zero on D
[I]=find(dvec==0);
dvec(I)=[];
V(:,I)=[];
NV = zeros(size(V));
%sort the elements of dvec in descending order according with the absolute value of dvec
[L,index_dv] = sort(abs(dvec));
index_dv = flipud(index_dv);
dvec=dvec(index_dv);
%insert the elements of dvec on the diagonal
ND=diag(dvec);
%sort the columns of V according with index_dv
NV=V(:,index_dv);
return
```

fld

```
% function [W,D]=fld(A,C)
%-----------------------------------------------------------------------------------------------
%
% Filename:            fld.m
% Thesis Advisor:      Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:    Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:              Diogo Pereira, Naval Postgraduate School, Monterey, CA, 2002
% Modified by:         Colin K Lee, Naval Postgraduate school, Monterey, CA, 2004
% Descriptions:        Function computes the Fisher Linear Discriminant W that
%                      maximizes the ratio |W^T S_B W|/|W^T S_W W|  where S_W is
%                      the within class  scatter matrix, and S_B is the between class scatter
%                      matrix
% Inputs:
%                      A:     data matrix, containing data in columns
%                      C:     row array containing numbers representing the classes of
%                              the elements in A
% Outputs:
%                      W:     weight matrix contains the vectors in columns
%                      D:     matrix containing the eigenvalues on the diagonal

% Function(s) called:  pca.m, sortem.m
%
%-----------------------------------------------------------------------------------------------
function [W,D]=fld(A,C);
MeanG=mean(A,2);%MeanG is the general mean of the data in A
Ad=A-MeanG*ones(1,size(A,2)); %Ad is matrix A minus the mean of A
ST=Ad*Ad';% ST is the total scatter matrix
clear Ad

Nclass=0; %Nclass counts the number of classes on A

%Compute the variance within all the
classes%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SW=zeros(size(A,1)); %SW is the within class scatter matrix % Produce sq matrix w/ all
zeros elements
while size(C,2)>0,
   w=C(1,1);
   [I]=find(C==w);
   MeanClass=mean(A(:,I),2);
   Ad=A(:,I)- MeanClass*ones(1,size(A(:,I),2));
   SW=SW+Ad*Ad'; %adds the within class scatter matrix of each class
   A(:,I)=[]; %eliminates the data already used
   C(I)=[];   %eliminates the number corresponding to the class from C vector
   Nclass=Nclass+1;
```

```
end
%-------------------------------------------------------------------------------------------------------
%used on debbuging
SW;
inv(SW)*SW;
cond(SW);
SB=ST-SW;
cond(SB);
cond(ST);
%-------------------------------------------------------------------------------------------------------

N=min(size(A,1), Nclass-1);%N will be the minimum between the dimension of the data
and Nclass-1
[W,EVA]=eigs(SB,SW,N);
W;
EVA;

%Set to zero the eigenvalues not equal to a finite value
[I]=find(isfinite(EVA)==1); %Obtain the index of the eigenvalues that are finite
EVAC=zeros(size(EVA));
EVAC(I)=EVA(I);
EVA=EVAC;%EVA will contain just the eigenvalues that are finite

% obtain index of eigenvalues greater than 0.001 times greater eigenvalue
[I]=find(abs(EVA)>(0.001*max(max(abs(EVA)))));
EVAC=zeros(size(EVA));
EVAC(I)=EVA(I);  %EVAC will have only values greater than a minimum value other-
wise the value is set to zero
[W,D]=sortem(W,EVAC);% order and eliminates eigenvector corresponding to zero ei-
genvalue

% this part normalizes W so that the norm of each column will be one
NW=ones(size(W,1),1)*sqrt(sum((W.^2),1));
W=W./NW;
return
```

pca_eig

```
%-----------------------------------------------------------------------------------------------------
%
% Filename:              pca_eig.m
% Thesis Advisor:      Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:  Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:                 Diogo Pereira, Naval Postgraduate School, Monterey, CA, 2002
% Modified by:          Colin K Lee, Naval Postgraduate school, Monterey, CA, 2004
% Descriptions:         Routine program exams the error rate vs. various number of
%                              eigenvectors used in PCA-based classification
% Inputs:
%                              A_all:  data matrix
%                              cnetroids_all:  centrodis computed from A_all
% Outputs:                Error Rate
% Function(s) called:  none
%
%-----------------------------------------------------------------------------------------------------
clear all
close all
clc

% Load the training images
load A_16
A=double(A_16);

% Compute the pca of the training images
[W,m,Amean,Ad]=pca(A);

% Compute the projection matrix P
P=W'*(Ad);

% Computing the centroid of each class
C=[];
C=meanclass(P,T,Person);

% % Load the testing images
load A_5
A=double(A_5);
Ad=A-Amean*ones(1,size(A,2));
% Compute the projection matrix P
P=W'*(Ad);

% Computing the error rate in function to the number of eigenvectors
dmin=100000000;
kmax=size(P,1);
```

```
N_images=size(P,2);
for k=1:kmax
   [L,M]=size(P);  %there are M objects
   [N,K]=size(C);  %there are K classes
    D=zeros(1,M);
        for i=1:M
            dif=C(1:k,:)-P(1:k,i)*ones(1,K);
            [dist,pos]=min(( sum((dif).^2,1) ));
                if dist<dmin
                   D(1,i)=pos;
                end
        end
   NZD=find(D~=0);
   D(NZD)=Person(D(NZD)); %Obtain the class number based on the position number
   error=T-D;
   n_error(k)=sum((error~=0),2);
end
figure
plot(n_error/N_images*100,'+b')
xlabel('Number of eigenvectors');
ylabel('Error rate (%)');
title('Error rate versus number of eigenvectors');
grid;
```

variation

```
%----------------------------------------------------------------------------------------------------
%
% Filename:           pca_eig.m
% Thesis Advisor:     Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:   Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:             Diogo Pereira, Naval Postgraduate School, Monterey, CA, 2002
% Modified by:        Colin K Lee, Naval Postgraduate school, Monterey, CA, 2004
% Descriptions:       Routine program determines distance between each image and
%                     all centroids using LDA.
% Inputs:
%                     A_all:  data matrix
% Outputs:            min,mean, and max distance of each image to the centroids.
% Function(s) called: pca.m, fld.m, meanclasse.m
%
%----------------------------------------------------------------------------------------------------

clc
clear all;
close all;

load A_all
A=double(A_all);
Person=[1 2 3 4 5 6 8 9 11 12 13 14 15 16 50 51 52 ...
      53 54 55 56 57 58 59 60 61 62 63 64 65 66 ...
      67 68 69 70 71 72 73 74 75 76 77 78 79 80 ...
      81 82 83 84 85];
% load centroids_all

[W,m,Amean,Ad,EVA]=pca(A);
% Compute the projection matrix P
P=W'*(Ad);
% Compute the projection matrix P obtained with the FLD(Fisher Linear Discriminant)
[Wopt,D]=fld(P,T);
P=Wopt'*P;

% Computing the centroid of each class
C=meanclass(P,T,Person);
Ad=A-Amean*ones(1,size(A,2));
P=W'*(Ad);
P=Wopt'*P;

% Classify the testing images
dmin=100000000;
ImageClassif=zeros(1,size(P,2));
```

```
[L,M]=size(P); %there are M objects
[N,K]=size(C);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=C-P(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin;
      D(1,i)=pos;
   end
end
NZD=find(D~=0);
D(NZD)=Person(D(NZD));

error=T-D;
%PersonError=Person(T(find(error~=0)))
PersonError=T(find(error~=0));
n_error=sum((error~=0),2);
Dist=distance(P,C);
DistWithLabelLDA=[0 Person;T' Dist];

wk1write('DistancesLDA.wk1',DistWithLabelLDA);
ff=wk1read('DistancesLDA.wk1');

save ff
```

kcv

```
%-----------------------------------------------------------------------------------------------------
%
% Filename:             kcv.m
% Thesis Advisor:       Prof. M.P. Fargues, Naval Postgraduate School, Monterey, CA
% Thesis CoAdvisor:  Prof. G. Karunasiri, Naval Postgraduate School, Monterey, CA
% Author:               Diogo Pereira, Naval Postgraduate School, Monterey, CA, 2002
% Modified by:          Colin K Lee, Naval Postgraduate school, Monterey, CA, 2004
% Descriptions:         Routine program tests the 2 schemes using k-fold cross validation
%                       6 training/4 testing. The number of elements in each class is left
%                       constant.
% Inputs:
%                       A_all:  data matrix
% Outputs:
%                       Error Rate in %
% Function(s) called: pca.m, fld.m, meanclass.m
%
%-----------------------------------------------------------------------------------------------------
clear all;
close all;
clc

% A_all will contain the training images
% B will contain the testing images
% T is the class number of the images on A
% TB is the class number of the images on B

load A_all
Atemp=double(A_all);
Ttemp=T;
N_trials=900;
Person = [1 2 3 4 5 6 8 9 11 12 13 14 15 16 50 51 52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85];
epct=0; % accumulator of the number of errors per class
epctW=0;
epctW1=0;
epctW2=0;
epctW3=0;
epct80=0;
epct80W1=0;
epct80W2=0;
epct80W3=0;
epct50=0;
```

```
epct50W1=0;
epct50W2=0;
epct50W3=0;
epct40W3=0;
epctLDA=0;

for j=1:N_trials
    A       =   Atemp;
    T       =   Ttemp;
    B       =   [];
    TB      =   [];
    Remove  =   [];
  % Generates testing and training sets
  for i=1:150  % i goes from 1 to the total number of pictures
     f=randperm(10);
     L1=(i-1)*10+1;
     L2=i*10;
     A(:,L1:L2)=A(:,(i-1)*10+f); % mix the samples in each class
     T(:,L1:L2)=T(:,(i-1)*10+f); % adjust the class numbers
     IndSamples=[L1 L1+1 L1+2 L1+3];
     B=[B A(:,IndSamples)];
     TB=[TB T(:,IndSamples)];
     Remove=[Remove IndSamples];
 end
 A(:,Remove)=[];
 T(:,Remove)=[];

% PCA

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Compute the pca of the training images A
 [W,m,Amean,Ad]=pca(A);
 % Compute the projection matrix P
 P=W'*(Ad);

   %%%% used to do crossvalidation directly without pca or lda
   PD=A;
   %AmeanD=mean(A,2);
   PD=A-Amean*ones(1,size(A,2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Compute the centroid of each class contained in Person
 C=meanclass(P,T,Person);
 CD=meanclass(PD,T,Person);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Using the testing images B
 Ad=B-Amean*ones(1,size(B,2));
 % Compute the projection matrix P
 P=W'*(Ad);
   %P(1,:)=[];
 % Classify the testing images
 dmin=100000000;

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % classifying the images directly

 PD=Ad;

 [L,M]=size(PD); %there are M objects
[N,K]=size(CD);  %there are K classes
D=zeros(1,M);
for i=1:M
    dif=CD-PD(:,i)*ones(1,K);
    [dist,pos]=min(( sum((dif).^2,1) ));
    if dist<dmin
        D(1,i)=pos;
    end
end
NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorDirect(j)=sum((error~=0),2);
 %computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epct=epct+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using all eigenvectors

 [ww,wh]=size(W);
 Neig=wh;
 Temp1=P(1:Neig,:);
 Temp2=C(1:Neig,:);
 [L,M]=size(Temp1); %there are M objects
```

```
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCAW(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epctW=epctW+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using all eigenvectors but the first
[ww,wh]=size(W);
Neig=wh;
Temp1=P(2:Neig,:);
Temp2=C(2:Neig,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCAW1(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epcp3=epc(:,101:150);
 epc=[epcp1;epcp2;epcp3];
 epc=sum(epc,1);
 epctW1=epctW1+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using all eigenvectors but the first two
[ww,wh]=size(W);
Neig=wh;
Temp1=P(3:Neig,:);
Temp2=C(3:Neig,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCAW2(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12, 50);
 epc=sum(epc,1);
 epctW2=epctW2+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using all eigenvectors but the first three
[ww,wh]=size(W);
Neig=wh;
Temp1=P(4:Neig,:);
Temp2=C(4:Neig,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
```

```
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCAW3(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epctW3=epctW3+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using 80 eigenvectors
Neig=80;
Temp1=P(1:Neig,:);
Temp2=C(1:Neig,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
    dif=Temp2-Temp1(:,i)*ones(1,K);
    [dist,pos]=min(( sum((dif).^2,1) ));
    if dist<dmin
        D(1,i)=pos;
    end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCA80(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12, 50);
 epc=sum(epc,1);
 epct80=epct80+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using 80 eigenvectors without the first eigenvectors
Neig=80;
Temp1=P(2:Neig+1,:);
Temp2=C(2:Neig+1,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
```

```
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCA80W1(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epct80W1=epct80W1+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using 80 eigenvectors without the first two eigenvectors
Neig=80;
Temp1=P(3:Neig+2,:);
Temp2=C(3:Neig+2,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCA80W2(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epct80W2=epct80W2+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using 80 eigenvectors without the first three eigenvectors
```

```
Neig=80;
Temp1=P(4:Neig+3,:);
Temp2=C(4:Neig+3,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCA80W3(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epct80W3=epct80W3+epc;

 %using 50 eigenvectors
Neig=50;
Temp1=P(1:Neig,:);
Temp2=C(1:Neig,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCA50(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epct50=epct50+epc;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using 50 eigenvectors without the first eigenvectors
Neig=50;
Temp1=P(2:Neig+1,:);
Temp2=C(2:Neig+1,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
    dif=Temp2-Temp1(:,i)*ones(1,K);
    [dist,pos]=min(( sum((dif).^2,1) ));
    if dist<dmin
        D(1,i)=pos;
    end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCA50W1(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epct50W1=epct50W1+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using 50 eigenvectors without the first two eigenvectors
Neig=50;
Temp1=P(3:Neig+2,:);
Temp2=C(3:Neig+2,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
    dif=Temp2-Temp1(:,i)*ones(1,K);
    [dist,pos]=min(( sum((dif).^2,1) ));
    if dist<dmin
        D(1,i)=pos;
    end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCA50W2(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epct50W2=epct50W2+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %using 50 eigenvectors without the first three eigenvectors
Neig=50;
Temp1=P(4:Neig+3,:);
Temp2=C(4:Neig+3,:);
[L,M]=size(Temp1); %there are M objects
[N,K]=size(Temp2);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=Temp2-Temp1(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
      D(1,i)=pos;
   end
end
 NZD=find(D~=0);
 D(NZD)=Person(D(NZD));
 error=TB-D;
 n_errorPCA50W3(j)=sum((error~=0),2);
%computes the number of errors per class
 errornz=(error~=0);
 epc=reshape(errornz,12,50);
 epc=sum(epc,1);
 epct50W3=epct50W3+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % LDA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Compute the pca of the training images
  N_pictures = 10;
  temp=(N_pictures-1)*length(Person);
  [W,m,Amean,Ad,EVA]=pca(A,temp); % This reduces the dimension to N-c = 36 -
6=30 the best result was with 29
  % Compute the projection matrix P
  P=W'*(Ad);
%   P(1,:)=[];
  % Compute the projection matrix P obtained with the FLD(Fisher Linear Discriminant)
  [Wopt,D]=fld(P,T);
  P=Wopt'*P;
  % Computing the centroid of each class
  C=meanclass(P,T,Person);
  % Using the testing data
  Ad=B-Amean*ones(1,size(B,2));
  P=W'*(Ad);
%   P(1,:)=[];
  P=Wopt'*P;
  % Classify the testing images
  dmin=100000000;
  [L,M]=size(P); %there are M objects
[N,K]=size(C);  %there are K classes
D=zeros(1,M);
for i=1:M
   dif=C-P(:,i)*ones(1,K);
   [dist,pos]=min(( sum((dif).^2,1) ));
   if dist<dmin
       D(1,i)=pos;
   end
end

  NZD=find(D~=0);
  D(NZD)=Person(D(NZD));
  error=TB-D;
  n_errorLDA(j)=sum((error~=0),2);
  %computes the number of errors per class
  errornz=(error~=0);
  epc=reshape(errornz,12,50);
  epc=sum(epc,1);
  epctLDA=epctLDA+epc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

N_samples=4*50*3; % we are using 6/4 cross validation
h1=figure;
L1=min(n_errorDirect);
L2=max(n_errorDirect);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorDirect,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
```

```
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h1,'CrossValidationDirect6_4.fig');

h2=figure;
L1=min(n_errorPCAW);
L2=max(n_errorPCAW);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCAW,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h1,'CrossValidation_all_eigen_6_4.fig');

h3=figure;
L1=min(n_errorPCAW1);
L2=max(n_errorPCAW1);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCAW1,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h3,'CrossValidationall_eigenW1_6_4.fig');

h4=figure;
L1=min(n_errorPCAW2);
L2=max(n_errorPCAW2);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCAW2,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h4,'CrossValidationall_eigenW2_6_4.fig');

h5=figure;
L1=min(n_errorPCAW3);
L2=max(n_errorPCAW3);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCAW3,temp1);
```

```
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h5,'CrossValidationall_eigenW3_6_4.fig');

h6=figure;
L1=min(n_errorPCA80);
L2=max(n_errorPCA80);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCA80,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h6,'CrossValidation80eig_6_4.fig');

h7=figure;
L1=min(n_errorPCA80W1);
L2=max(n_errorPCA80W1);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCA80W1,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h7,'CrossValidation80W1eig_6_4.fig');

h8=figure;
L1=min(n_errorPCA80W2);
L2=max(n_errorPCA80W2);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCA80W2,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h8,'CrossValidation100W2eig_6_4.fig');

h9=figure;
L1=min(n_errorPCA80W3);
L2=max(n_errorPCA80W3);
```

```
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCA80W3,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h9,'CrossValidation80W3eig_6_4.fig');

h10=figure;
L1=min(n_errorPCA50);
L2=max(n_errorPCA50);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCA50,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h10,'CrossValidation50eig_6_4.fig');

h11=figure;
L1=min(n_errorPCA50W1);
L2=max(n_errorPCA50W1);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCA50W1,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h11,'CrossValidation50W1eig_6_4.fig');

h12=figure;
L1=min(n_errorPCA50W2);
L2=max(n_errorPCA50W2);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCA50W2,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h12,'CrossValidation50W2eig_6_4.fig');

h13=figure;
```

```
L1=min(n_errorPCA50W3);
L2=max(n_errorPCA50W3);
temp1=linspace(0,L2,L2+1);
N1=hist(n_errorPCA50W3,temp1);
plot(temp1/N_samples*100,N1/N_trials*100,'bd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h13,'CrossValidation50W3eig_6_4.fig');

h14=figure;
L1=min(n_errorLDA);
L2=max(n_errorLDA);
temp2=linspace(0,L2,L2+1);
N2=hist(n_errorLDA,temp2);
plot(temp2/N_samples*100,N2/N_trials*100,'rd-');
title('Histogram of the number of errors');
xlabel('Percentage of errors');
ylabel('Percentage of the simulations');
grid;
saveas(h14,'CrossValidationLDA_6_4.fig');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% saving the error per class
ns=((N_samples/50)*N_trials)/100;
epct=epct/ns;
epctW=epctW/ns;
epctW=epctW1/ns;
epctW=epctW2/ns;
epctW=epctW3/ns;
epct80=epct80/ns;
epct80W1=epct80W1/ns;
epct80W2=epct80W2/ns;
epct80W3=epct80W3/ns;
epct50=epct50/ns;
epct50W1=epct50W1/ns;
epct50W2=epct50W2/ns;
epct50W3=epct50W3/ns;
epctLDA=epctLDA/ns;
M=[Person;epct;epctW;epctW1;epctW2;epctW3;epct80;epct80W1;epct80W2;epct80W3;
epct50;epct50W1;epct50W2;epct50W3;epctLDA];

%t='Errors per class 60%-train-40%test';
filename1='ErrorPerClass6_4perc';
```

```
%wk1write(filename,t,1,1);
%wk1write(filename,Person,3,2);
wk1write(filename1,M);
dd1=wk1read(filename1);
save dd1

ns=100/((N_samples/50)*N_trials);
epct=epct/ns;
epctW=epctW/ns;
epctW1=epctW1/ns;
epctW2=epctW2/ns;
epctW3=epctW3/ns;
epct80=epct80/ns;
epct80W1=epct80W1/ns;
epct80W2=epct80W2/ns;
epct80W3=epct80W3/ns;
epct50=epct50/ns;
epct50W1=epct50W1/ns;
epct50W2=epct50W2/ns;
epct50W3=epct50W3/ns;
epctLDA=epctLDA/ns;
M=[Person;epct;epctW;epctW1;epctW2;epctW3;epct80;epct80W1;epct80W2;epct80W3;
epct50;epct50W1;epct50W2;epct50W3;epctLDA];

%t='Errors per class 60%-train-40%test';
filename2='ErrorPerClass6_4abs';
%wk1write(filename,t,1,1);
%wk1write(filename,Person,3,2);
wk1write(filename2,M);
dd2=wk1read(filename1);
save dd2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

APPENDIX B. SIMULATION RESULTS

This appendix contains simulations results obtained from k-fold cross validation implementation for different numbers of class combinations. All simulations used 200 iterations. For example, "Mean and Median Error Rate with 14 Classes and 200 Iterations" represents mean and median error rates using 14 classes for PCA and LDA-based schemes.

Mean and Median Error Rate with 14 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	45.58	18.04	14.04	14.83	20.13	23.58	0.00
2	38.04	10.33	14.92	12.00	10.29	12.50	0.00
3	17.00	27.58	15.21	9.42	23.42	34.33	2.71
4	14.83	0.00	0.00	0.00	0.00	0.13	0.00
5	0.04	31.50	31.67	19.42	10.63	22.75	0.00
6	18.13	14.04	13.88	4.33	5.50	8.46	2.08
8	11.96	21.88	22.46	16.38	15.04	20.75	0.58
9	0.33	0.00	0.21	0.25	2.92	21.46	0.04
11	13.38	11.38	10.42	5.83	4.08	6.88	0.58
12	3.63	4.25	3.79	4.58	5.75	12.50	0.00
13	76.88	33.75	35.67	25.04	31.33	21.04	0.96
14	5.58	0.33	2.13	0.63	1.96	4.54	0.00
15	0.88	10.88	6.25	3.17	0.58	1.17	0.17
16	4.04	0.75	1.75	3.21	8.83	10.21	1.50
mean	**17.88**	**13.19**	**12.31**	**8.51**	**10.03**	**14.31**	**0.62**
median	**12.67**	**11.13**	**12.15**	**5.21**	**7.29**	**12.50**	**0.10**

Mean and Median Error Rate with 15 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	44.79	16.67	12.54	12.42	17.33	19.13	0.00
2	36.96	17.50	20.46	18.17	7.71	9.21	0.00
3	15.88	23.04	18.33	13.08	16.83	24.04	1.46
4	15.29	0.00	0.00	0.04	0.04	0.04	0.00
5	0.00	13.33	19.13	14.79	14.58	13.83	0.29
6	17.79	8.38	11.04	1.29	5.17	8.29	1.42
8	12.04	22.13	22.71	16.63	15.92	24.25	1.21
9	0.50	0.00	1.08	0.83	2.21	6.17	0.00
11	12.58	11.88	7.21	5.13	3.33	7.83	0.67
12	3.04	3.00	2.50	3.33	7.13	6.17	0.04
13	77.13	48.71	43.54	30.92	35.96	22.75	0.33
14	4.50	0.00	0.04	0.29	0.75	2.67	0.00
15	1.00	3.96	2.79	1.38	0.71	0.75	0.08
16	4.21	0.58	1.17	2.25	5.88	5.58	0.75
50	0.00	0.29	3.04	3.13	2.67	31.29	0.00
mean	**16.38**	**11.30**	**11.04**	**8.24**	**9.08**	**12.13**	**0.42**
median	**12.04**	**8.38**	**7.21**	**3.33**	**5.88**	**8.29**	**0.08**

Mean and Median Error Rate with 16 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	45.29	24.08	16.25	17.71	14.79	21.63	0.00
2	38.63	22.79	18.46	19.67	21.04	12.88	0.00
3	12.96	15.13	15.42	20.46	24.96	25.17	1.08
4	14.83	0.04	0.00	0.00	0.00	0.08	0.00
5	0.00	13.33	19.00	20.17	9.38	9.00	0.33
6	17.17	7.21	9.75	8.29	4.71	9.00	1.00
8	12.96	22.71	25.04	21.46	17.29	23.46	1.46
9	0.75	0.00	0.21	1.38	0.96	1.21	0.00
11	13.13	11.63	7.42	12.54	5.75	5.83	0.71
12	3.46	3.25	2.08	4.08	1.83	4.46	0.04
13	77.33	59.00	45.42	41.25	36.42	20.04	0.04
14	4.63	0.04	0.04	0.00	0.00	2.42	0.00
15	1.63	4.46	1.63	4.17	0.42	0.83	0.00
16	5.54	0.33	1.25	4.33	6.33	7.38	0.71
50	0.58	6.17	12.75	12.42	15.04	22.29	0.04
51	0.88	6.46	7.04	2.96	7.67	11.58	0.00
mean	**15.61**	**12.29**	**11.36**	**11.93**	**10.41**	**11.08**	**0.34**
median	**9.25**	**6.83**	**8.58**	**10.35**	**7.00**	**9.00**	**0.04**

Mean and Median Error Rate with 17 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	45.67	34.83	26.63	29.63	18.04	23.92	0.00
2	38.79	21.96	19.42	24.17	18.42	13.38	0.00
3	15.96	11.13	17.29	30.71	32.25	33.79	1.08
4	15.54	0.54	0.00	0.00	0.25	0.79	0.00
5	0.04	16.46	24.13	23.08	16.58	12.71	0.50
6	16.92	5.54	15.00	14.13	6.13	8.08	0.08
8	13.67	21.79	23.21	27.33	16.92	23.63	1.46
9	0.50	0.00	0.00	1.50	0.96	1.21	0.00
11	14.75	11.67	6.50	11.54	5.08	4.33	0.54
12	3.75	3.38	3.38	7.08	3.83	5.00	0.00
13	77.50	67.25	44.46	57.46	37.00	23.00	0.63
14	4.08	0.08	0.04	0.33	0.17	1.67	0.00
15	0.71	2.92	2.13	3.79	1.17	1.50	0.00
16	4.17	1.13	3.29	6.75	6.71	7.33	0.67
50	0.25	0.92	3.50	9.46	9.63	19.63	0.00
51	0.54	2.71	4.38	3.75	4.17	11.88	0.00
52	0.00	9.33	14.96	25.25	20.63	17.08	0.04
mean	**14.87**	**12.45**	**12.25**	**16.23**	**11.64**	**12.29**	**0.29**
median	**4.17**	**5.54**	**6.50**	**11.54**	**6.71**	**11.88**	**0.00**

Mean and Median Error Rate with 18 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	46.79	31.33	20.46	29.67	19.42	24.08	0.00
2	38.67	28.29	13.42	26.04	21.00	13.71	0.00
3	15.79	10.96	15.79	31.54	31.17	32.96	1.46
4	15.33	0.58	0.00	0.00	0.33	0.92	0.00
5	0.00	14.25	24.25	20.08	15.92	12.83	0.46
6	16.79	5.79	13.46	14.54	5.67	6.92	0.17
8	12.71	21.29	23.08	27.29	17.71	25.17	1.79
9	0.63	0.00	0.00	3.38	0.67	0.83	0.00
11	13.38	11.50	6.92	12.04	4.17	4.08	0.29
12	3.08	2.63	2.50	6.21	3.13	4.88	0.00
13	77.67	70.13	47.54	56.29	37.25	23.83	0.42
14	5.08	0.17	0.13	0.33	0.00	1.54	0.00
15	1.08	4.13	3.88	5.33	1.29	1.38	0.04
16	3.71	0.96	3.58	8.21	6.46	6.88	0.75
50	4.17	4.92	5.75	13.25	11.88	22.29	0.00
51	0.79	1.33	2.83	5.25	5.04	11.46	0.00
52	0.00	9.29	12.50	22.46	21.25	15.54	0.00
53	0.25	18.79	32.71	32.00	30.67	34.79	5.04
mean	**14.22**	**13.13**	**12.71**	**17.44**	**12.94**	**13.56**	**0.58**
median	**4.63**	**7.54**	**9.71**	**13.90**	**9.17**	**12.15**	**0.02**

Mean and Median Error Rate with 19 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	44.92	28.83	17.25	26.92	19.00	23.75	0.08
2	39.17	34.50	16.17	26.08	23.92	14.71	0.00
3	14.54	9.75	17.38	33.75	30.33	31.96	1.08
4	16.83	1.13	0.00	0.04	1.08	1.17	0.00
5	0.00	10.33	24.63	23.58	14.54	12.63	0.33
6	17.50	5.96	12.21	15.08	6.21	8.46	0.04
8	12.17	20.88	22.46	26.63	20.13	26.79	1.42
9	0.38	0.00	0.00	1.63	0.92	1.25	0.00
11	14.79	12.33	7.29	8.79	4.38	4.00	0.54
12	3.88	3.08	2.79	6.42	4.71	5.92	0.00
13	77.46	72.00	47.63	54.17	36.13	23.29	0.46
14	5.58	0.54	0.33	0.92	0.08	4.17	0.00
15	0.88	3.54	10.17	9.38	5.08	4.79	0.00
16	3.88	1.42	1.88	6.79	6.38	7.08	0.42
50	4.13	5.63	6.50	16.08	11.21	25.54	0.00
51	0.92	0.88	2.38	7.92	7.13	12.42	0.00
52	0.00	5.75	7.83	16.71	18.08	16.71	0.00
53	8.00	25.79	30.79	31.71	27.88	35.04	8.29
54	0.00	0.33	0.79	1.29	5.67	9.13	0.08
mean	**13.95**	**12.77**	**12.02**	**16.52**	**12.78**	**14.15**	**0.67**
median	**5.58**	**5.75**	**7.83**	**15.08**	**7.13**	**12.42**	**0.04**

Mean and Median Error Rate with 20 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	45.88	29.33	18.29	26.83	19.17	24.58	0.08
2	37.79	36.21	16.08	27.71	18.38	12.92	0.04
3	15.08	9.71	17.83	34.42	28.96	30.00	1.21
4	16.38	1.63	0.00	0.29	0.38	0.71	0.00
5	0.00	11.58	28.50	26.92	16.96	14.13	0.33
6	16.71	5.33	11.54	14.04	5.04	6.71	0.13
8	11.54	20.38	22.79	30.54	21.88	27.50	1.42
9	0.42	0.00	0.00	0.54	0.38	0.63	0.00
11	13.25	12.46	8.83	9.13	3.96	4.29	0.58
12	3.38	3.04	2.88	6.58	4.46	6.00	0.00
13	76.42	76.46	52.25	62.83	35.75	22.46	0.08
14	5.17	0.21	0.13	1.17	0.25	4.67	0.00
15	1.33	7.08	12.71	10.38	5.38	3.75	0.00
16	5.29	1.96	2.21	8.71	7.29	7.88	0.79
50	4.33	6.33	7.46	15.92	14.08	27.71	0.00
51	1.50	1.46	3.04	7.38	6.92	11.38	0.00
52	0.00	5.00	6.04	18.38	20.42	16.67	0.00
53	6.04	25.21	30.50	32.92	29.42	33.71	7.33
54	0.00	0.33	0.63	2.33	6.54	11.92	0.04
55	0.46	13.08	19.83	27.25	30.21	32.25	3.13
mean	**13.05**	**13.34**	**13.08**	**18.21**	**13.79**	**14.99**	**0.76**
median	**5.23**	**6.71**	**10.19**	**14.98**	**10.69**	**12.42**	**0.06**

Mean and Median Error Rate with 25 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	47.13	33.75	26.25	33.75	25.67	27.63	0.00
2	39.29	37.38	17.96	26.79	18.71	11.13	0.21
3	17.63	12.58	13.08	27.08	17.33	16.58	2.92
4	15.50	2.04	0.00	0.00	0.08	0.00	0.00
5	0.00	15.50	31.88	30.17	22.42	16.08	0.50
6	17.08	4.50	12.00	10.88	4.50	5.13	0.25
8	13.38	25.25	30.42	41.04	31.25	35.04	3.75
9	0.92	0.04	0.29	0.75	0.42	0.79	0.00
11	14.67	17.42	10.71	11.04	3.21	3.50	0.71
12	3.79	3.00	5.04	6.83	1.04	3.21	0.08
13	77.67	83.04	57.42	63.83	42.83	24.42	0.25
14	6.04	0.83	1.33	4.17	0.29	3.79	0.00
15	1.17	8.13	7.50	8.88	1.46	0.79	0.00
16	4.58	2.08	1.63	4.00	6.42	6.75	0.33
50	4.21	7.71	9.38	14.54	17.71	29.21	0.71
51	1.92	0.96	2.08	4.46	6.96	14.29	0.08
52	0.04	11.88	11.58	16.42	19.21	14.79	0.17
53	7.96	28.83	32.71	34.63	31.46	35.46	11.00
54	0.25	1.67	2.04	2.25	10.04	16.50	0.00
55	6.83	21.79	35.79	36.92	36.04	44.04	2.83
56	3.21	6.46	7.13	15.88	11.96	16.63	2.79
57	1.29	12.17	14.63	19.46	19.46	14.42	2.25
58	23.00	28.00	32.71	38.21	32.17	41.50	6.88
59	3.58	9.71	12.17	12.71	11.29	13.58	3.83
60	11.25	28.21	38.21	40.29	30.50	36.33	9.38
mean	**12.90**	**16.12**	**16.56**	**20.20**	**16.10**	**17.26**	**1.96**
median	**6.04**	**11.88**	**12.00**	**15.88**	**17.33**	**14.79**	**0.33**

Mean and Median Error Rate with 30 Classes and 200 Iterations:

Clas #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	47.54	33.54	21.88	33.79	26.67	28.33	0.33
2	39.13	37.13	21.75	28.42	16.38	14.17	1.96
3	20.38	17.75	21.58	27.38	16.54	19.71	3.38
4	16.58	2.38	0.00	0.00	0.04	0.00	0.00
5	0.00	16.17	36.29	30.17	26.00	17.17	0.25
6	18.71	5.29	12.63	10.96	5.54	7.67	0.04
8	11.38	26.42	36.21	41.50	33.67	37.08	3.88
9	0.75	0.08	0.04	1.17	0.08	1.08	0.21
11	16.96	22.38	17.29	17.25	4.25	5.08	1.38
12	3.83	2.96	5.29	7.00	1.04	2.08	0.00
13	77.92	83.33	61.42	59.83	39.46	25.29	0.58
14	5.33	1.71	2.08	3.96	0.71	3.08	0.00
15	1.42	10.25	8.67	8.25	1.67	1.13	0.00
16	4.67	2.29	2.71	4.67	7.38	7.75	1.17
50	7.17	10.71	17.08	20.67	27.17	41.17	0.83
51	2.13	1.33	2.00	3.79	7.29	15.13	0.08
52	0.00	9.71	9.75	14.88	21.42	8.04	0.29
53	20.67	29.83	33.21	34.54	32.38	37.17	9.96
54	1.71	8.08	9.17	9.00	11.83	21.42	0.17
55	7.46	34.21	37.83	37.88	36.04	34.17	2.21
56	3.38	5.92	5.58	15.21	14.13	16.71	2.92
57	9.54	19.13	24.00	27.29	18.71	12.96	2.25
58	32.17	31.96	37.63	42.71	39.42	45.92	10.00
59	14.33	17.50	18.17	21.13	19.04	20.54	4.88
60	16.92	38.67	43.33	45.63	36.96	34.63	7.04
61	4.13	6.67	9.75	14.88	7.17	33.67	1.42
62	9.92	14.63	16.25	18.25	17.83	17.75	10.75
63	7.21	39.13	41.38	61.29	50.67	56.75	4.58
64	18.88	38.08	45.21	47.42	31.13	36.00	3.71
65	22.58	14.67	22.33	24.67	23.29	46.13	3.71
mean	**14.76**	**19.40**	**20.68**	**23.78**	**19.13**	**21.59**	**2.60**
median	**9.73**	**15.42**	**17.73**	**20.90**	**18.27**	**18.73**	**1.40**

Mean and Median Error Rate with 35 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	48.46	35.33	31.21	35.63	32.96	33.79	1.42
2	40.63	38.96	26.38	31.25	19.79	15.17	1.54
3	18.92	15.33	19.58	34.04	17.58	18.46	3.21
4	17.54	3.88	0.00	0.00	0.00	0.00	0.00
5	0.04	16.50	33.04	30.54	25.33	20.08	0.25
6	17.96	5.67	12.08	12.00	6.21	5.75	0.00
8	13.33	26.67	36.04	39.25	30.25	34.21	4.38
9	0.63	0.00	0.04	0.79	0.17	0.96	0.17
11	16.67	24.67	19.63	18.88	4.17	4.83	0.88
12	4.25	3.54	6.00	6.83	0.63	2.25	0.13
13	77.67	86.88	63.88	65.46	38.21	26.75	0.71
14	5.58	1.58	2.29	3.58	0.04	3.21	0.00
15	1.92	13.63	9.33	7.33	0.92	1.08	0.00
16	4.33	2.75	3.17	9.42	7.92	8.75	0.75
50	7.63	12.54	18.46	25.71	24.54	43.29	2.04
51	2.58	2.58	4.42	9.29	9.96	14.54	0.00
52	0.04	10.25	9.67	14.04	14.79	7.83	0.33
53	24.88	35.75	37.96	39.33	35.33	39.13	13.58
54	1.67	8.46	10.38	10.71	12.58	26.58	0.46
55	14.29	39.00	40.00	46.42	40.83	39.42	2.13
56	9.46	6.08	7.67	15.67	17.13	25.83	2.75
57	10.88	24.96	25.92	29.46	22.71	11.79	2.79
58	33.96	35.38	42.92	47.92	48.54	51.67	8.00
59	15.17	17.96	17.29	22.25	20.79	21.88	6.58
60	23.29	40.00	46.29	57.13	39.83	37.25	6.79
61	4.79	8.29	9.92	13.04	8.33	29.25	0.92
62	9.58	14.63	17.58	20.29	19.33	17.92	10.04
63	10.21	36.92	37.46	58.71	50.25	57.79	5.13
64	26.42	41.00	53.63	58.00	38.29	40.17	10.13
65	25.88	15.00	24.96	27.54	23.71	49.29	5.75
66	15.04	15.67	22.54	53.75	57.33	60.54	5.50
67	10.29	20.42	23.50	27.33	22.17	25.71	2.00
68	43.17	54.21	44.50	50.29	43.04	45.46	20.38
69	26.21	31.25	44.21	42.83	46.50	47.83	4.83
70	24.00	31.08	32.71	31.96	13.71	13.96	2.38
mean	**17.35**	**22.19**	**23.85**	**28.48**	**22.68**	**25.21**	**3.60**
median	**14.29**	**16.50**	**22.54**	**27.54**	**20.79**	**25.71**	**2.04**

Mean and Median Error Rate with 40 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	48.21	34.63	28.58	33.54	30.67	32.50	3.75
2	39.29	37.21	25.25	32.25	21.88	18.38	1.75
3	17.04	13.83	28.42	31.17	25.63	20.58	2.96
4	16.54	2.96	0.00	0.00	0.25	0.00	0.00
5	0.04	17.54	37.00	32.50	24.08	20.29	0.17
6	19.75	6.75	12.83	11.33	7.83	7.58	0.00
8	14.54	27.58	40.04	38.38	31.29	35.92	4.71
9	0.75	0.04	0.00	0.75	0.33	0.58	0.13
11	16.25	27.17	20.08	20.96	5.67	5.92	1.88
12	4.21	3.88	4.54	6.38	0.71	2.54	0.04
13	78.21	86.83	64.50	62.96	37.46	26.29	0.50
14	5.17	2.21	2.75	4.54	0.17	4.25	0.00
15	1.58	12.79	10.96	8.92	0.96	1.21	0.04
16	5.04	3.08	5.63	8.71	7.79	8.29	0.96
50	7.88	13.79	20.75	27.08	26.00	46.83	2.08
51	2.79	3.04	6.42	7.63	12.71	18.71	0.17
52	0.08	10.46	10.13	13.58	12.00	8.17	0.50
53	28.46	38.96	40.92	38.92	36.75	38.92	16.00
54	2.08	10.33	13.04	13.79	16.25	29.42	0.63
55	15.88	40.29	39.83	39.38	40.21	41.71	1.79
56	9.58	6.92	5.83	12.79	15.96	24.75	3.33
57	11.25	27.29	25.38	30.42	23.00	13.96	3.17
58	34.17	36.58	47.42	50.13	51.96	54.58	9.79
59	14.92	17.50	19.21	20.71	19.79	20.25	5.21
60	32.04	45.63	51.96	57.83	41.42	37.83	7.75
61	4.21	6.58	6.83	10.38	7.00	28.67	0.63
62	10.17	16.25	18.46	20.21	20.79	20.75	11.00
63	13.92	39.79	45.75	66.46	52.50	58.71	5.33
64	27.25	41.13	42.79	53.88	37.83	40.54	10.92
65	28.46	19.63	28.08	28.96	28.04	51.21	8.83
66	17.79	17.38	28.50	55.08	59.79	61.92	5.71
67	9.38	20.00	25.29	30.08	20.50	24.50	2.46
68	43.08	56.75	46.92	48.21	42.08	46.96	20.33
69	30.71	35.13	44.08	42.71	48.96	51.88	5.96
70	25.08	31.46	32.08	28.04	12.38	13.04	1.92
71	20.79	27.71	25.29	28.88	29.08	29.42	1.17
72	18.83	31.25	40.46	34.88	29.71	33.33	7.08
73	58.63	52.58	46.04	54.04	47.96	43.33	7.54
74	30.29	18.71	25.38	28.17	32.92	41.17	3.42
75	5.63	9.58	22.88	31.29	34.17	38.21	3.63
mean	**19.25**	**23.78**	**26.01**	**29.15**	**24.86**	**27.58**	**4.08**
median	**16.06**	**19.17**	**25.33**	**29.52**	**24.85**	**27.48**	**2.71**

Mean and Median Error Rate with 45 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	47.58	33.17	26.75	32.92	30.08	31.63	6.21
2	39.75	38.88	30.75	35.17	16.75	16.54	6.08
3	17.33	15.83	32.92	25.54	24.46	17.88	3.38
4	17.54	3.83	0.04	0.00	0.21	0.04	0.00
5	0.00	17.00	31.29	28.83	26.79	23.25	0.00
6	18.50	5.83	9.33	9.25	8.04	8.46	0.13
8	12.25	25.88	39.46	34.25	31.88	36.63	4.13
9	0.79	0.00	0.00	0.38	0.33	0.71	0.25
11	16.75	26.38	21.79	22.46	5.50	7.08	0.96
12	4.25	4.13	4.46	7.33	1.00	5.08	0.21
13	79.29	87.58	68.50	68.46	40.00	27.92	0.63
14	5.50	1.75	1.67	3.88	0.00	3.54	0.00
15	2.21	13.08	14.38	12.79	1.75	1.04	0.00
16	4.00	2.42	6.04	7.25	6.75	7.04	0.25
50	8.58	12.29	18.38	23.13	25.63	46.63	3.21
51	2.42	2.71	6.38	5.92	10.50	16.58	0.17
52	0.08	11.58	9.92	14.25	15.17	12.38	0.67
53	33.04	40.17	42.13	38.83	36.63	38.88	14.58
54	1.88	8.83	11.54	12.33	14.75	29.67	0.50
55	16.46	43.29	39.13	36.54	39.54	42.67	1.71
56	15.58	11.63	7.75	13.17	18.25	29.46	3.83
57	13.00	27.13	26.25	29.50	23.54	17.00	4.79
58	35.08	39.46	44.79	47.83	49.75	53.08	9.54
59	14.29	16.25	20.71	22.04	21.25	21.67	6.58
60	33.75	49.88	49.96	50.04	42.50	38.00	12.75
61	5.25	9.75	8.42	12.50	9.63	33.79	0.67
62	13.00	18.46	22.38	23.92	24.54	23.38	12.33
63	18.21	41.17	46.00	64.04	50.71	59.42	6.96
64	31.63	42.67	44.08	46.46	36.58	43.79	11.42
65	27.67	17.29	21.67	26.63	28.21	53.83	5.58
66	21.92	25.17	35.50	55.83	60.38	61.58	6.29
67	17.25	24.00	27.42	31.50	27.54	29.92	2.50
68	55.38	63.50	45.42	45.38	43.25	49.75	18.92
69	34.71	37.42	44.21	42.88	50.92	54.71	4.08
70	25.42	32.79	32.29	20.83	10.96	12.25	2.67
71	19.83	27.08	25.38	29.79	29.50	29.92	0.92
72	21.83	32.67	42.50	35.67	28.63	35.88	9.50
73	58.71	53.04	50.08	53.54	46.29	42.63	7.63
74	36.88	20.13	29.08	30.92	33.46	41.96	3.13
75	5.21	13.50	22.42	28.33	32.67	35.88	3.79
76	31.75	44.29	37.38	55.00	63.29	65.38	8.08
77	24.88	40.83	45.04	58.83	55.25	72.63	10.50
78	18.13	18.54	21.96	12.42	12.38	12.96	9.67
79	30.38	37.50	31.96	33.96	48.88	63.33	6.79
80	28.83	29.83	31.63	34.13	35.75	37.25	3.25
mean	**21.48**	**25.97**	**27.31**	**29.44**	**27.11**	**30.96**	**4.78**
median	**18.13**	**25.17**	**27.42**	**29.50**	**27.54**	**29.92**	**3.79**

Mean and Median Error Rate with 50 Classes and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	47.71	34.46	28.67	34.33	32.88	33.71	9.54
2	39.58	38.88	31.00	36.25	18.04	17.96	6.25
3	19.58	18.79	35.92	30.96	28.75	23.25	2.58
4	18.46	4.75	0.13	0.21	0.33	0.00	0.00
5	0.00	16.67	26.33	26.13	27.17	22.83	0.04
6	19.08	6.00	7.38	7.54	8.96	9.04	0.17
8	13.63	27.25	37.13	32.71	30.79	33.88	4.04
9	0.38	0.13	0.00	0.13	0.04	0.13	0.63
11	16.71	26.75	22.83	23.08	4.58	5.67	0.67
12	3.75	3.71	3.54	6.63	0.88	3.92	0.17
13	80.21	87.96	69.42	68.71	38.50	32.21	2.38
14	5.17	4.79	3.25	4.21	0.21	5.29	0.04
15	2.00	13.75	13.63	12.13	1.50	0.33	0.04
16	4.75	2.79	6.79	8.88	9.33	9.96	0.92
50	9.25	15.17	19.71	26.25	28.04	49.25	4.29
51	3.17	3.21	5.96	5.83	11.75	19.67	0.71
52	4.46	9.75	8.29	13.38	13.38	9.92	1.50
53	32.38	38.13	40.08	37.25	35.42	38.75	14.92
54	1.92	10.38	12.71	13.71	16.50	30.08	0.50
55	16.92	44.96	40.00	35.38	42.13	42.04	1.00
56	17.88	14.83	9.25	19.33	23.29	32.21	4.17
57	17.71	29.04	26.25	29.83	24.75	19.63	4.71
58	34.42	39.25	43.92	46.13	49.58	54.17	11.88
59	14.67	16.58	18.79	21.79	20.21	20.54	6.04
60	41.79	55.08	55.04	52.29	48.71	42.46	12.54
61	6.29	10.54	8.33	13.08	11.00	31.50	1.50
62	12.25	17.29	20.96	22.38	22.67	20.46	11.46
63	17.29	37.96	42.54	63.96	48.88	59.33	5.63
64	34.38	45.33	45.29	44.96	36.67	43.96	11.63
65	28.21	19.63	22.29	28.58	34.21	52.54	3.96
66	19.29	23.25	41.83	65.50	69.83	67.42	6.96
67	15.29	22.21	25.58	30.79	26.33	29.83	0.79
68	53.79	62.21	42.67	44.08	42.54	49.46	18.50
69	33.63	35.96	44.21	42.96	54.04	55.58	6.13
70	27.96	34.79	34.00	20.04	12.25	15.75	2.88
71	20.33	27.08	24.67	30.33	30.13	30.83	3.13
72	23.54	33.21	37.63	33.33	27.75	34.21	9.08
73	58.04	53.29	51.46	54.83	48.29	45.13	7.71
74	37.29	19.25	27.13	29.92	32.54	42.04	3.00
75	8.96	19.67	27.75	30.83	36.00	37.04	3.38
76	36.00	47.92	43.13	56.92	68.54	75.83	8.92
77	24.75	41.04	43.58	58.50	57.79	68.04	8.58
78	20.13	21.83	21.96	12.38	12.63	16.67	10.38
79	31.75	39.54	33.29	34.17	49.00	64.79	4.88
80	26.13	27.29	28.42	31.00	32.54	34.17	3.79
81	38.58	35.25	37.25	49.08	53.08	53.38	16.54
82	12.50	17.17	14.42	14.46	10.54	24.88	1.17
83	32.46	41.08	37.17	42.75	48.88	51.88	17.38
84	15.96	10.21	14.92	23.17	27.29	33.58	9.54
85	19.25	26.79	38.79	29.75	29.50	33.13	2.75
mean	**22.39**	**26.66**	**27.51**	**30.02**	**28.77**	**32.57**	**5.39**
median	**19.25**	**26.66**	**27.51**	**30.02**	**28.75**	**32.57**	**4.04**

Mean and Median Error Rate with 15 Classes, without Class #13, and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	46.71	24.21	15.13	20.29	14.08	19.21	0.00
2	36.63	23.17	19.21	23.46	21.96	12.17	0.00
3	15.88	16.83	18.79	28.08	27.08	28.71	1.79
4	15.42	0.04	0.00	0.00	0.00	0.04	0.00
5	0.00	10.96	17.67	25.29	10.29	12.46	0.04
6	16.50	5.58	9.33	9.42	3.67	9.67	0.71
8	10.58	17.96	19.21	25.50	17.13	20.88	0.29
9	0.75	0.00	0.29	2.33	0.88	1.88	0.00
11	14.42	12.25	7.42	12.42	6.92	7.92	0.63
12	3.54	3.17	1.83	4.25	2.42	5.25	0.04
14	4.38	0.00	0.00	0.33	0.13	3.13	0.00
15	1.17	3.50	1.58	3.42	0.25	1.75	0.00
16	5.17	0.38	1.46	5.58	6.38	7.96	1.33
50	0.46	6.46	12.58	15.71	15.17	23.75	0.04
51	0.96	6.54	7.13	6.58	5.33	15.58	0.00
mean	**11.50**	**8.74**	**8.77**	**12.18**	**8.78**	**11.36**	**0.33**
median	**7.87**	**6.50**	**8.10**	**10.80**	**6.65**	**10.51**	**0.04**

Mean and Median Error Rate with 20 Classes, without Class #13, and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	45.75	32.42	21.46	27.50	18.75	21.92	0.13
2	37.42	35.04	17.79	28.71	21.88	12.75	0.00
3	14.63	9.67	17.63	27.50	16.79	20.92	1.79
4	16.38	1.75	0.00	0.00	0.13	0.17	0.00
5	0.00	10.04	26.75	26.50	16.04	17.13	0.00
6	16.63	5.17	10.17	12.38	4.42	4.75	0.00
8	10.58	19.42	21.75	29.33	21.46	26.21	1.17
9	0.42	0.04	0.04	0.58	0.17	0.46	0.00
11	13.17	13.25	6.96	9.08	4.25	4.67	0.58
12	3.42	2.79	3.08	5.92	4.17	4.79	0.00
14	4.58	0.21	0.08	1.04	0.13	3.63	0.00
15	1.42	9.50	11.67	10.96	4.83	3.71	0.04
16	4.63	2.08	2.00	7.83	6.08	6.58	0.17
50	4.00	6.04	6.88	16.58	13.88	27.96	0.00
51	0.88	0.88	1.50	7.83	6.50	13.21	0.00
52	0.08	14.58	15.04	22.08	21.46	20.13	0.08
53	8.21	24.50	33.08	35.17	31.75	36.63	8.29
54	0.00	0.13	0.17	2.58	6.71	12.83	0.00
55	1.13	11.17	19.88	27.46	25.83	33.13	3.00
56	0.13	4.79	5.13	10.71	9.25	10.92	0.46
mean	**11.05**	**10.17**	**11.05**	**15.49**	**11.72**	**14.12**	**0.79**
median	**4.29**	**7.77**	**8.56**	**11.67**	**7.98**	**12.79**	**0.02**

Mean and Median Error Rate with 25 Classes, without Class #13, and 200 Iterations:

Class #	PCA50	PCA50W1	PCA50W2	PCA50W3	PCA50W4	PCA50W5	LDA
1	45.42	34.29	24.67	33.96	26.29	24.54	0.04
2	38.25	37.25	20.75	27.00	21.46	13.88	0.46
3	15.75	11.63	13.29	26.58	18.88	17.17	2.21
4	16.08	2.33	0.00	0.00	0.04	0.00	0.00
5	0.00	14.29	29.29	27.29	19.58	13.96	0.08
6	18.63	6.04	11.75	11.88	4.08	5.33	0.00
8	12.13	23.71	29.25	40.04	29.42	35.33	2.04
9	0.79	0.00	0.17	0.38	0.25	0.96	0.00
11	15.33	17.33	11.42	11.42	2.92	3.33	0.33
12	3.83	2.79	5.75	7.46	1.75	1.29	0.04
14	7.25	1.08	0.92	3.83	0.38	4.00	0.00
15	1.33	8.63	7.08	7.21	0.83	0.50	0.00
16	4.08	3.00	2.04	4.42	6.67	7.96	0.79
50	4.88	9.29	9.13	18.54	20.75	37.29	0.33
51	2.08	1.54	2.54	4.79	6.50	17.50	0.08
52	0.04	8.92	8.29	15.54	16.96	10.17	0.33
53	8.13	25.08	28.42	32.67	30.33	35.29	8.63
54	0.21	1.75	2.17	3.83	11.25	17.46	0.00
55	6.96	17.92	34.71	37.88	33.42	37.67	3.08
56	2.21	5.21	5.29	16.46	12.58	18.08	2.21
57	2.04	12.58	17.63	23.42	22.08	20.46	2.79
58	22.58	29.17	32.67	37.96	32.00	41.25	5.13
59	2.83	8.54	10.83	13.75	11.25	14.79	3.58
60	11.58	25.17	35.42	39.71	27.67	33.13	6.38
61	4.67	7.79	12.38	14.21	10.08	30.04	0.83
mean	**9.88**	**12.61**	**14.23**	**18.41**	**14.70**	**17.66**	**1.57**
median	**4.88**	**8.92**	**11.42**	**15.54**	**12.58**	**17.17**	**0.33**

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] D.C. Pereira, "Face Recognition Using Infrared Imaging," Electrical Engineer Thesis, Naval Postgraduate School, Monterey, California, December 2002.

[2] "History of Infrared Technology," FLIR Systems, Inc., [http://www. flirthermography.com/about/ir_history.asp], last accessed April 2004.

[3] "Infrared Basics," Infrared Solutions, [http://www.infraredsolutions.com/html/technology/infBasicsF.shtml], last accessed April 2004.

[4] "How Do Infrared Cameras Work," FLIR Systems, Inc., [http://www. flirthermography.com/about/how_infrared_cameras.asp], last accessed April 2004.

[5] "Thermal IR imaging, military FLIR," Sierra Pacific Corp., [http://www.x20.org/library/thermal/desert_fox.htm], last accessed April 2004.

[6] "FIREFIGHTING," Raytheon Thermal-Eye, [http://www.raytheoninfrared.com/firefighting/], last accessed April 2004.

[7] "Camera Technology," Infrared Solutions, [http://www.infraredsolutions.com/html/technology/cameraTechF.shtml], last accessed April 2004.

[8] Infrared Solutions Inc. Specification, Rev. F 11/01, *IR 160 Data Sheet*, November 2004.

[9] M. P. Fargues, Thesis Topics Presentation in EC3000 (Introduction to Graduate Research), Naval Postgraduate School, Monterey, California, 2003 (unpublished).

[10] K.A. Bloch, Specification, Rev. A 1_104219B, *Manual, Communications, Imager, Full,* Infrared Solutions Inc., Plymouth, Minnesota, January 2002.

[11] "Principal Component Analysis," M. Cairns, [http://www.dcs.gla.ac.uk/~mc/1stYearReport/2.6_PCA.htm], last accessed Apr 15th 2004.

[12] B.S. Everitt and G. Dunn, *Applied Multivariate Data Analysis*, Oxford University Press, London, 1992.

[13] "PCA-Principal Component Analysis," Maintenance Engineering, The University of Manchester, [http://www.eng.man.ac.uk/mech/merg/research/ datafsion.org.uk/ pca. Html], last accessed April 2004.

[14] L.N. Trefethen, and Bau III, D., *Numerical Linear Algebra*, SIAM, Philadelphia, Pennsylvania, 1997.

[15] W.S. Yambor, *Analysis of PCA-based and Fisher Discriminant-Based Image Recognition Algorithms*, Technical Report CS-00-103, pp. 11-12, Colorado State University, Fort Collins, Colorado, July 2000.

[16] C.W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[17] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 711-720, July 1997.

[18] X. Chen, P.J. Flynn, and K.W. Bowyer, "PCA-Based Face Recognition in Infrared Imagery: Baseline and Comparative Studies," *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, Nice, France, October 2003.

[19] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, Second Edition, John Wiley & Sons, New York, New York, 2001.

[20] R. Gutierrez-Osuna, "Linear Discriminant Analysis," Texas A&M University, Kingsville, Texas, [http://research.cs.tamu.edu/prism/lectures/pr/pr_l10.pdf], last accessed April 2004.

[21] [http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-12.html], last accessed April 2004.

[22] R. Gutierrez-Osuna, "Validation," Texas A&M University, Kingsville, Texas, [http://research.cs.tamu.edu/prism/lectures/iss/iss_l13.pdf], last accessed April 2004.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. John P. Powers
   Chairman, ECE Department
   Naval Postgraduate School
   Monterey, California

4. Prof. Monique P. Fargues
   ECE Department
   Naval Postgraduate School
   Monterey, California

5. Prof. Gamani Karunasiri
   Physics Department
   Naval Postgraduate School
   Monterey, California

6. Colin K. Lee
   SUSHIP San Diego Det. Pearl
   Pearl Harbor, Hawaii